ANALISIS PERBANDINGAN PROSES PENGOLAHAN CITRA MENGGUNAKAN FPGA DAN MIKROKOMPUTER

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:
Muhammad Naufal
NIM: 145150300111025

PROGRAM STUDI TEKNIK KOMPUTER
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

ANALISIS PERBANDINGAN PROSES PENGOLAHAN CITRA MENGGUNAKAN FPGA DAN MIKROKOMPUTER

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun Oleh :
Muhammad Naufal
NIM: 145150300111025

Skripsi ini telah diuji dan dinyatakan lulus pada
6 Juli 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Wijaya Kurniawan, S.T, M.T.	Dahnial Syauqy, S.T., M.T., M.Sc.
NIP. 19820125 201504 1 002	NIK. 2016078704231002

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D
NIP. 197105182003121001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 28 Mei 2018

Muhammad Naufal

NIM: 145150300111025

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah Subhanallahu wa Ta'ala karena berkah rahmat dan hidayah-Nya, penulis dapat menyelesaikan penelitian dan Laporan Skripsi untuk memperoleh gelar Sarjana Teknik yang berjudul **Analisis Perbandingan Proses Pengolahan Citra Menggunakan FPGA dan Mikrokomputer**

Dalam pelaksanaan dan penyusunan laporan skripsi ini, tidaklah sedikit hambatan dan kesulitan yang penulis hadapi. Namun, penulis menyadari bahwa kelancaran dalam penyusunan laporan ini tidak lain adalah berkat doa, bantuan dan bimbingan dari berbagai pihak, sehingga kendala-kendala yang penulis hadapi dapat teratasi. Penghargaan dan terima kasih yang sebesar-besarnya penulis sampaikan kepada:

1. Bunda Haryeni Ratna Dewi, Ayah Tamsil Makmur Zubaedah selaku kedua orang tua dan saudara-saudari kandung dari penulis yang tidak henti memberikan doa, semangat dan bantuan materi kepada penulis.
2. Putri Ayu Delina Sari, wanita yang memotivasi dan mengisi hati dan hari penulis saat mengerjakan penelitian
3. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D selaku ketua jurusan Teknik Informatika periode 2016 – saat ini.
4. Bapak Sabriansyah Rizqika Akbar, S.T., M. Eng. selaku ketua program studi Teknik Komputer periode 2016 – saat ini.
5. Bapak Wijaya Kurniawan, S.T, M.T. selaku dosen pembimbing I yang selalu membantu memberikan bimbingan dan arahan selama pengerjaan skripsi.
6. Bapak Dahnial Syauqy, S.T., M.T., M.Sc. selaku dosen pembimbing II yang selalu membantu memberikan bimbingan dan arahan selama pengerjaan skripsi.
7. Haqqi Rizqi S.T., selaku dosen pembimbing III yang selalu membantu memberikan bimbingan dan arahan selama pengerjaan skripsi.

8. Muhamad Taufiq Firmansyah, Gusti Arief Gilang, dan Nafisa yang selalu mengisi hari-hari penulis dan membuat penulis sedikit tidak waras sehingga penulis tetap semangat dalam menyelesaikan penulisan skripsi ini.
9. Teman-teman PSDM HIMATEKKOM, serta teman-teman Teknik Komputer 2014 yang selalu memberikan motivasi kepada penulis untuk menyelesaikan penulisan skripsi ini.
10. Teman-Teman grup Anak Sholeh, yang selalui menyediakan konten-konten bermanfaat saat penulis melakukan penelitian
11. Seluruh pihak yang tidak dapat disebutkan satu persatu yang telah berperan dalam penyelesaian skripsi ini.

Akhir kata penulis menyadari bahwa dalam penulisan skripsi ini masih terdapat banyak kekurangan yang perlu diperbaiki. Karena itu, kritik dan saran yang membangun sangat diharapkan oleh penulis. Semoga kedepannya skripsi ini dapat bermanfaat bagi semua pihak yang membacanya.

Malang, 28 Mei 2018

Muhammad Naufal
lafuan@hotmail.com

ABSTRAK

Pengolahan citra merupakan aspek penting dalam kehidupan karena kebutuhan manusia semakin hari semakin bertambah, sehingga dibutuhkan sistem yang mampu mengolah citra dengan efektif. Untuk mengetahui sistem apa yang bisa mengolah citra secara efektif, dilakukan analisis perbandingan pengolahan citra antara kedua sistem, FPGA dan Mikrokomputer. FPGA yang digunakan adalah myRIO sedangkan Mikrokomputer yang digunakan adalah Raspberry Pi. Penelitian dilakukan dengan algoritma pengolahan citra yang umum seperti *Gaussian blur, Laplacian edge* dan *Sobel edge*. Pertama citra RGB dikonversi ke *grayscale* agar citra mudah diolah lalu dihilangkan noisenya dengan algoritma *gaussian blur.* Setelah itu citra dideteksi tepi dengan algoritma *laplacian edge* dan *sobel edge*. Pengujian dilakukan dengan mengolah tiga ukuran citra berbeda pada tiga algoritma yang berbeda dan dilakukan sepuluh kali pengujian dan diambil waktu rata-rata pengolahan citra pada kedua sistem. Hasil waktu rata-rata pengolahan citra dengan algoritma *Gaussian blur* adalah 0.485s pada FPGA dan 0.165s pada Mikrokomputer. Untuk algortitma *Laplacian edge* waktu rata-rata adalah 0.492s pada FPGA dan 0.202s pada Mikrokomputer sedangkan untuk algoritma *Sobel edge* waktu rata-rata nya adalah 0.498s pada FPGA dan 0.234s pada Mikrokomputer. Namun sebenarnya untuk semua algoritma, waktu FPGA tetap sama namun berbeda pada tiga citra yang berukuran berbeda masing-masing adalah 0.01053, 0.03074dan 0.06076 detik.

Kata Kunci: Citra, FPGA, Mikrokomputer*, Gaussian Blur, Laplacian Edge, Sobel Edge*

ABSTRACT

Image processing is an important aspect in life because human needs are increasingly growing day by day, so we need a system that can process image effectively. To find out what systems can effectively process images, a comparison analysis of image processing between two systems, FPGA and Microcomputer is performed. FPGA used is myRIO while the Microcomputer used is Raspberry Pi. The study was conducted with common image processing algorithms such as Gaussian blur, Laplacian edge and Sobel edge. First the RGB image is converted to grayscale for easy image processing and then eliminated its noisenya with gaussian blur algorithm. After that the image is detected edge with laplacian edge and sobel edge algorithms. The test was performed by processing three different image sizes on three different algorithms and performed ten tests and taken the average time of image processing on both systems. The mean time of image processing with Gaussian blur algorithm is 0.485s on FPGA and 0.165s on Microcomputer. For Laplacian edge mean time algorithm is 0.492s on FPGA and 0.202s on Microcomputer while for Sobel edge algorithm its average time is 0.498s on FPGA and 0.234s on Microcomputer. But actually for all algorithms, FPGA time remains the same but different on three different sized images respectively are 0.01053, 0.03074 and 0.06076 seconds.

Keywords: *Image, FPGA, Microcomputer, Gaussian Blur, Laplacian Edge, Sobel Edge*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

ANALISIS PERBANDINGAN PROSES PENGOLAHAN CITRA MENGGUNAKAN FPGA DAN MIKROKOMPUTER

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun oleh:
Muhammad Naufal
NIM: 145150300111025

PROGRAM STUDI TEKNIK KOMPUTER
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

ANALISIS PERBANDINGAN PROSES PENGOLAHAN CITRA MENGGUNAKAN FPGA DAN MIKROKOMPUTER

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Teknik

Disusun Oleh :
Muhammad Naufal
NIM: 145150300111025

Skripsi ini telah diuji dan dinyatakan lulus pada
6 Juli 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Wijaya Kurniawan, S.T, M.T.	Dahnial Syauqy, S.T., M.T., M.Sc.
NIP. 19820125 201504 1 002	NIK. 2016078704231002

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T., M.T., Ph.D
NIP. 197105182003121001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 28 Mei 2018

Muhammad Naufal

NIM: 145150300111025

KATA PENGANTAR

Puji syukur penulis panjatkan kepada Allah Subhanallahu wa Ta'ala karena berkah rahmat dan hidayah-Nya, penulis dapat menyelesaikan penelitian dan Laporan Skripsi untuk memperoleh gelar Sarjana Teknik yang berjudul **Analisis Perbandingan Proses Pengolahan Citra Menggunakan FPGA dan Mikrokomputer**

Dalam pelaksanaan dan penyusunan laporan skripsi ini, tidaklah sedikit hambatan dan kesulitan yang penulis hadapi. Namun, penulis menyadari bahwa kelancaran dalam penyusunan laporan ini tidak lain adalah berkat doa, bantuan dan bimbingan dari berbagai pihak, sehingga kendala-kendala yang penulis hadapi dapat teratasi. Penghargaan dan terima kasih yang sebesar-besarnya penulis sampaikan kepada:

1. Bunda Haryeni Ratna Dewi, Ayah Tamsil Makmur Zubaedah selaku kedua orang tua dan saudara-saudari kandung dari penulis yang tidak henti memberikan doa, semangat dan bantuan materi kepada penulis.
2. Putri Ayu Delina Sari, wanita yang memotivasi dan mengisi hati dan hari penulis saat mengerjakan penelitian
3. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D selaku ketua jurusan Teknik Informatika periode 2016 – saat ini.
4. Bapak Sabriansyah Rizqika Akbar, S.T., M. Eng. selaku ketua program studi Teknik Komputer periode 2016 – saat ini.
5. Bapak Wijaya Kurniawan, S.T, M.T. selaku dosen pembimbing I yang selalu membantu memberikan bimbingan dan arahan selama pengerjaan skripsi.
6. Bapak Dahnial Syauqy, S.T., M.T., M.Sc. selaku dosen pembimbing II yang selalu membantu memberikan bimbingan dan arahan selama pengerjaan skripsi.
7. Haqqi Rizqi S.T., selaku dosen pembimbing III yang selalu membantu memberikan bimbingan dan arahan selama pengerjaan skripsi.

8. Muhamad Taufiq Firmansyah, Gusti Arief Gilang, dan Nafisa yang selalu mengisi hari-hari penulis dan membuat penulis sedikit tidak waras sehingga penulis tetap semangat dalam menyelesaikan penulisan skripsi ini.
9. Teman-teman PSDM HIMATEKKOM, serta teman-teman Teknik Komputer 2014 yang selalu memberikan motivasi kepada penulis untuk menyelesaikan penulisan skripsi ini.
10. Teman-Teman grup Anak Sholeh, yang selalui menyediakan konten-konten bermanfaat saat penulis melakukan penelitian
11. Seluruh pihak yang tidak dapat disebutkan satu persatu yang telah berperan dalam penyelesaian skripsi ini.

Akhir kata penulis menyadari bahwa dalam penulisan skripsi ini masih terdapat banyak kekurangan yang perlu diperbaiki. Karena itu, kritik dan saran yang membangun sangat diharapkan oleh penulis. Semoga kedepannya skripsi ini dapat bermanfaat bagi semua pihak yang membacanya.

Malang, 28 Mei 2018

Muhammad Naufal
lafuan@hotmail.com

ABSTRAK

Pengolahan citra merupakan aspek penting dalam kehidupan karena kebutuhan manusia semakin hari semakin bertambah, sehingga dibutuhkan sistem yang mampu mengolah citra dengan efektif. Untuk mengetahui sistem apa yang bisa mengolah citra secara efektif, dilakukan analisis perbandingan pengolahan citra antara kedua sistem, FPGA dan Mikrokomputer. FPGA yang digunakan adalah myRIO sedangkan Mikrokomputer yang digunakan adalah Raspberry Pi. Penelitian dilakukan dengan algoritma pengolahan citra yang umum seperti *Gaussian blur, Laplacian edge* dan *Sobel edge*. Pertama citra RGB dikonversi ke *grayscale* agar citra mudah diolah lalu dihilangkan noisenya dengan algoritma *gaussian blur.* Setelah itu citra dideteksi tepi dengan algoritma *laplacian edge* dan *sobel edge*. Pengujian dilakukan dengan mengolah tiga ukuran citra berbeda pada tiga algoritma yang berbeda dan dilakukan sepuluh kali pengujian dan diambil waktu rata-rata pengolahan citra pada kedua sistem. Hasil waktu rata-rata pengolahan citra dengan algoritma *Gaussian blur* adalah 0.485s pada FPGA dan 0.165s pada Mikrokomputer. Untuk algortitma *Laplacian edge* waktu rata-rata adalah 0.492s pada FPGA dan 0.202s pada Mikrokomputer sedangkan untuk algoritma *Sobel edge* waktu rata-rata nya adalah 0.498s pada FPGA dan 0.234s pada Mikrokomputer. Namun sebenarnya untuk semua algoritma, waktu FPGA tetap sama namun berbeda pada tiga citra yang berukuran berbeda masing-masing adalah 0.01053, 0.03074dan 0.06076 detik.

Kata Kunci: Citra, FPGA, Mikrokomputer*, Gaussian Blur, Laplacian Edge, Sobel Edge*

ABSTRACT

Image processing is an important aspect in life because human needs are increasingly growing day by day, so we need a system that can process image effectively. To find out what systems can effectively process images, a comparison analysis of image processing between two systems, FPGA and Microcomputer is performed. FPGA used is myRIO while the Microcomputer used is Raspberry Pi. The study was conducted with common image processing algorithms such as Gaussian blur, Laplacian edge and Sobel edge. First the RGB image is converted to grayscale for easy image processing and then eliminated its noisenya with gaussian blur algorithm. After that the image is detected edge with laplacian edge and sobel edge algorithms. The test was performed by processing three different image sizes on three different algorithms and performed ten tests and taken the average time of image processing on both systems. The mean time of image processing with Gaussian blur algorithm is 0.485s on FPGA and 0.165s on Microcomputer. For Laplacian edge mean time algorithm is 0.492s on FPGA and 0.202s on Microcomputer while for Sobel edge algorithm its average time is 0.498s on FPGA and 0.234s on Microcomputer. But actually for all algorithms, FPGA time remains the same but different on three different sized images respectively are 0.01053, 0.03074 and 0.06076 seconds.

Keywords: *Image, FPGA, Microcomputer, Gaussian Blur, Laplacian Edge, Sobel Edge*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab landasan kepustakaan berisi uraian dan pembahasan tentang teori, konsep, model, metode, atau sistem dari literatur ilmiah, yang berkaitan dengan tema, masalah, atau pertanyaan penelitian. Dalam landasan kepustakaan terdapat landasan teori dari berbagai sumber pustaka yang terkait dengan teori dan metode yang digunakan dalam penelitian.

2.1 Tinjauan Pustaka

Berikut ini beberapa penelitian terkait sistem pengolahan citra pada FPGA dan Raspberry Pi yang menjadi landasan dan referensi dari sistem yang akan dibuat

Penelitian pertama yang dilakukan Maleeha Kiran, Kan Mei War, Lim Mei Kuan, Liang Kim Meng & Lai Weng Kin berjudul *"Implementing image processing algorithms using 'Hardware in the loop' approach for Xilinx FPGA"* dengan menggunakan pendekatan *Hardware in the loop* untuk penelitiannya. Penelitian ini bertujuan untuk menginvestigasi performa terbaik untuk sistem pengawasan otomatis dari berbagai *platform.* Sistem yang dibuat menggunakan aplikasi Matlab dan *library* dari *Xilinx System Generator* dengan metode *Illumination Analysis.* Penelitian ini membandingkan waktu pengolahan antara FPGA, C++ dan Matlab. Dari hasil penelitian, FPGA memiliki waktu pengolahan yang lebih cepat dibandingkan C++ dan Matlab. Namun menurut penulis penelitian ini kurang dapat dihandalkan karena masih sebatas *prototype* dan tidak di aplikasikan pada sistem yang sesungguhnya (Kiran, War, Kuan, Meng, & Kin, 2008)

Penelitian kedua yang dilakukan Megah Mulya dan Abdiansah berjudul "Penerapan Multi-threading untuk Meningkatkan Kinerja Pengolahan Citra Digital" dengan metode Multi-threading sebagai pendekatannya. Penelitian ini bertujuan membandingan waktu pengolahan citra digital dengan metode *multi-threading* dan *single-threading* yang biasa dikerakan didalam processor. Hasil penelitian ini adalah metode *multi-threading* lebih cepat dalah pengolahan citra digital. Hal ini disebabkan karena teknik pemrograman multi-threading dapat lebih mengoptimalkan kinerja prosesor terutama untuk komputer dengan prosesor ganda. Namun menurut penulis penelitian ini kurang dapat dihandalkan karena hasil pengujian perangkat lunak tersebut bisa berbeda untuk waktu pengujian yang berbeda. Hal ini disebabkan beban prosesor yang berbeda untuk situasi perangkat lunak aplikasi yang dieksekusi yang sedang dieksekusi oleh sistem operasi. (Mulya & Abdiansah, 2013)

Berdasarkan penelitian yang dilakukan pada penelitian pertama dan kedua, penulis mengambil parameter waktu dan nilai pixel pada citra setelah diproses dan juga mengambil mengambil kelebihan-kekurangan penelitian satu dan dua untuk membuat sistem baru yang lebih baik. Pada penelitian yang pertama, penggunaan *Simulink* pada *Matlab* dan *Xilinx System Generator* untuk pengolahan citra digital pada FPGA dirasa kurang pas karena clock simulasi pada

Simulink dan clock FPGA tidak sama, hal ini menyebabkan waktu pengolahan citra bergantung pada kecepatan clock Simulink di PC dan kecepatan clock FPGA. Sehingga waktu pengolahannya kurang tepat. Pada penelitian yang kedua, penggunaan PC sebagai alat uji yang menurut penulis kurang efisien dirubah dengan menggunakan platform yaitu myRIO-1900 dan Raspberry Pi model 3. Penggunaan myRIO-1900 alat uji dikarenakan pengembangan koding secara real-time OS saat melakukan read dan write dari dan untuk prosesor. Program yang dijalankan menggunakan real-time OS mempunyai performa yang lebih konsisten dibandingkan dengan sistem yang berjalan pada windows OS, hal tersebut dikarenakan window OS membagi prosesor untuk melakukan beberapa task seperti antivirus dan lainya. Sedangkan Raspberry Pi digunakan karena memilki bentuk yang compact, hemat energy dan salah satu mikrokomputer yang populer. Cocok untuk pengolahan citra di berbagai bidang

2.2 Dasar Teori

Dasar teori membahas teori yang diperlukan untuk menyusun penelitian yang dikerjakan.

2.2.1 Metode Grayscale

Suatu citra grayscale adalah suatu citra yang hanya memiliki warna tingkat keabuan. Penggunaan citra grayscale dikarenakan membutuhkan sedikit informasi yang diberikan pada tiap piksel dibandingkan dengan citra berwarna. Warna abu-abu pada citra grayscale adalah warna R (Red), G (Green), B (Blue) yang memiliki intensitas yang sama. Sehingga dalam grayscale image hanya membutuhkan nilai intensitas tunggal dibandingkan dengan citra berwarna membutuhkan tiga intensitas untuk tiap pikselnya. Intensitas dari citra grayscale disimpan dalam 8 bit integer yang memberikan 256 kemungkinan yang mana dimulai dari level 0 sampai dengan 255 (0 untuk hitam dan 255 untuk putih dan nilai diantaranya adalah derajat keabuan).

2.2.2 Gaussian Blur

Gaussian Blur berfungsi untuk mereduksi noise pada citra. Cara kerja *Gaussian Blur* adalah menghilangkan komponen high-frequency dari Gambar, sehingga teknik *Gaussian* dikatakan sebagai *low-pass filter*. (Abda'i, 2016) *Gaussian Blur* menggunakan fungsi distribusi Gaussian. Pada umumnya, *Gaussian Blur* direpresentasikan dalam bentuk array dua dimensi di [x, y] Filter Gaussian dituliskan dengan persamaan berikut:

$$G(x,y) = \frac{1}{2\pi\sigma x^2} e^{\frac{x^2+y^2}{2\sigma^2}} \qquad \textbf{(2.1)}$$

G(x,y) : elemen matriks Gauss pada posisi [x, y]

σ : standar deviasi atau sigma, semakin besar σ maka lokalisasi (jarak antar piksel) lemah atau auh, tapi untuk deteksi tepi semakin bagus. Semakin kecil σ maka lokalisasi (jarak antar piksel) bagus tapi untuk deteksi tepi semakin lemah

(x,y) : ukuran matriks Gauss yang menjangkau titik – x sampai + x , dan = titik tengahnya berada di x = 0 dan y = 0

2.2.3 Laplacian Edge

Lapacian Filter digunkan untuk mendeteksi edge (garis) dari sebuah citra. Lapacian Filter menggunakan fungsi turunan orde dua. (Prasetiyo, 2015). Metode ini akan mendeteksi *zero crossing* untuk menentukan garis atas antara hitam dan putih, yang terdapat pada turunan kedua citra tersebut. Nilai dari laplacian dapat di dekati dengan fungsi turunan orde satu dengan persamaan berikut:

$$\boldsymbol{f''(x) \cong f'(x) - f'(x+1)} \qquad \textbf{(2.2)}$$

Fungsi turunan orde satu dapat didekati dengan fungsi biasa, sehingga dari rumus diatas dapat didapat rumus fungsi sebagai berikut:

$$\boldsymbol{f''(x) \cong -f(x) + 2f(x+1) - f(x+2)} \qquad \textbf{(2.3)}$$

Sehingga dapat diperoleh matrik konvolusi sebagai berikut:

$$[-1 \quad 2 \quad -1] \qquad \textbf{(2.4)}$$

Ketika kita mengkombinasikan matrik fungsi turunan orde dua untuk baris dan untuk kolom bisa didapatkan :

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \qquad \textbf{(2.5)}$$

2.2.4 Sobel Edge

Metode Sobel merupakan pengembangan metode robert dengan menggunakan filter HPF yang diberi satu angka nol penyangga. (Maharani, Apriyana, Puspasari, & Angreni, 2007). Metode ini mengambil prinsip dari fungsi laplacian dan gaussian yang dikenal sebagai fungsi untuk membangkitkan HPF. Kelebihan dari metode sobel ini adalah kemampuan untuk mengurangi noise sebelum melakukan perhitungan deteksi tepi. Filter ini mendeteksikeseluruhan edge yang ada. Dalam prosesnya filterini menggunakan sebuah operator, yang dinamakan operator sobel. Operator sobel menggunakanmatriks NxN dengan berordo 3 x 3, 5 x 5, 7 x 7,dan sebagainya. Matriks seperti ini digunakanuntuk mempermudah untuk mendapatkan piksel tengah sehingga menjadi titik tengah matrik.Piksel tengah ini merupakan piksel yang akandiperiksa. Cara pemanfaatan matrik ini samaseperti pemakaian sebuah grid, yaitu dengan cara memasukkan piksel-piksel disekitar yang sedangdiperiksa (piksel tengah) ke dalam matrik. Cara yang demikian disebut spatial filtering. Teknik spatial filtering menggunkansebuah matrik tambahan yang di sebut *mask*. Ukuran matrik *mask* sama besar dengan matrik piksel yaitu NxN. Didalam *mask* ini intinya disimpan jenis operasi yang akan dilakukanterhadap matrik piksel, akan tetapi tidak semua filter spatial filtering menggunkan *mask* untuk menyimpan operasinya. Sobel operator diterapkan pada dalam dua buah *mask*, yaitu yang dapat dilihat pada Gambar 2.1 berikut :

-1	0	1
-2	0	2
-1	0	1

Mask (a) Vertikal

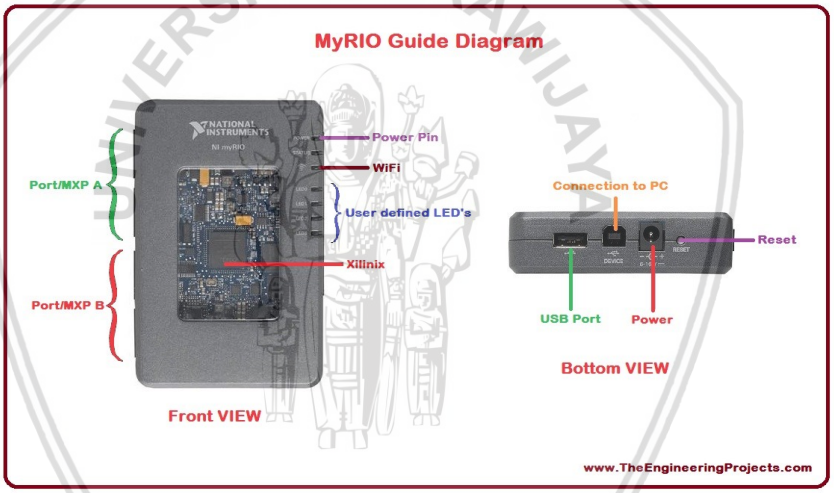
-1	-2	-1
0	0	0
1	2	1

Mask (b) Horizontal

Gambar 2.1 *Mask* Sobel 3 x 3

Sumber (Dokumen Penulis)

2.2.5 FPGA NI myRIO-1900



Gambar 2.2 NI myRIO-1900

Sumber (www.TheEngineeringProjects.com)

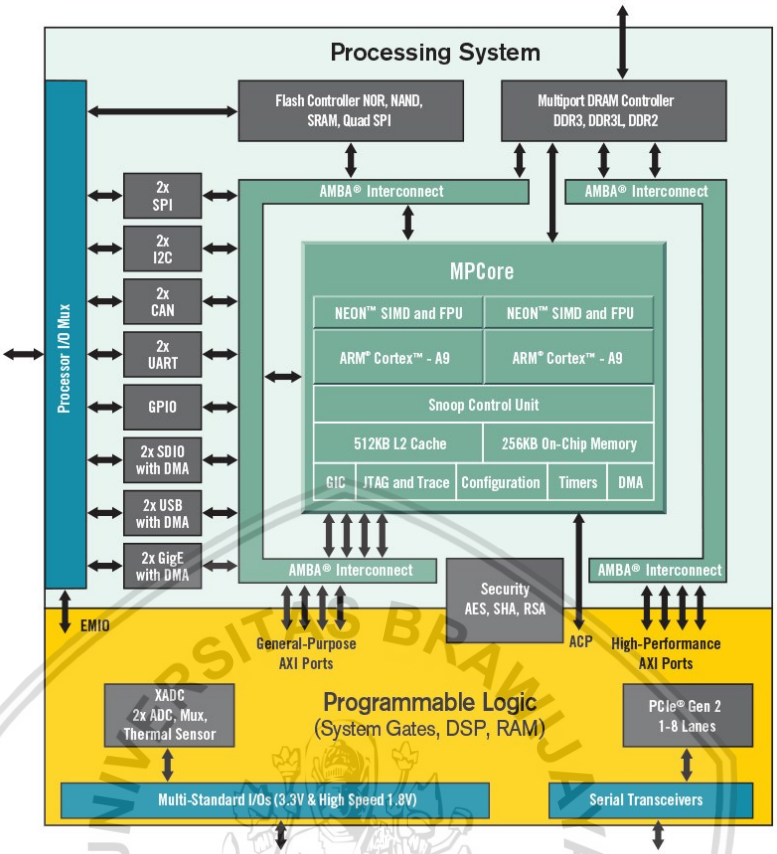
Gambar 2.2 adalah Gambar dari mikrokontroler NI myRIO, yang merupakan produk perangkat keras yang dibuat oleh perusahaan National Instruments. digunakan untuk memanipulasi fungsinya untuk membuat berbagai sistem. NI myRIO dilengkapi *dual-core* ARM® Cortex ™ -A9 , pengolahan real-time dan Xilinx FPGA yang I/O-nya disesuaikan untuk pembelajaran. NI myRIO menggunakan LabView sebagai IDE untuk mengakses fitur-fitur pada NI myRIO. Pada board NI my RIO terdapat beberapa periperal yang umum yang dapat digunakan seperti *bluetooth*, wifi, LED, *accelerometer*, push button, analog input dan output, serta RAM. Dapat juga dihubungkan dengan sebuah integrated circuit untuk membuat sistem yang lebih kompleks, atau dihubungkan ke board lain yang memang diperlukan dalam membuat sebuah sistem. Pada Tabel 2.1 dapat dilihat spesifikasi myRIO-1900.

Tabel 2.1 Spesifikasi NI myRIO

Power supply range	6-16VDC
Volt MXP Connector	0 – 5V
USB device port	USB 2.0 Hi-Speed
Volt MSP Connector	±10V
Processor	Xilinx Z-7010
Processor Speed	667 MHz
Processor Core	512 MB
FPGA type	Xilinx Z-7010
DDR 3 memory	256 MB
Resolution Analog Input	12 bits

2.2.6 Xilinx Zynq-7010

Xilinx Zynq-7010 adalah Chip FPGA yang ada di NI myRIO-1900. Xilinx Zynq-7010 dilengkapi dengan prosesor ARM Cortex-A9 dual-core yang terintegrasi dengan logika programmable 28nm Artix-7 untuk kinerja-per-watt yang sangat baik dan fleksibilitas desain maksimum. Dengan sel-sel logika hingga 6.6M dan ditawarkan dengan transceiver mulai dari 6.25Gb / s hingga 12.5Gb / s, perangkat Zynq-7000 memungkinkan desain yang sangat terdiferensiasi untuk berbagai aplikasi embedded termasuk sistem bantuan driver multi-kamera dan 4K2K Ultra-HDTV . (Xilinx, 2017). Pada Gambar 2.3 dapat dilihat blok diagram dari Xilinx Zynq-7010



Gambar 2.3 Blok Diagram Xilinx Zynq-7010

Sumber: (www.xilinx.com)

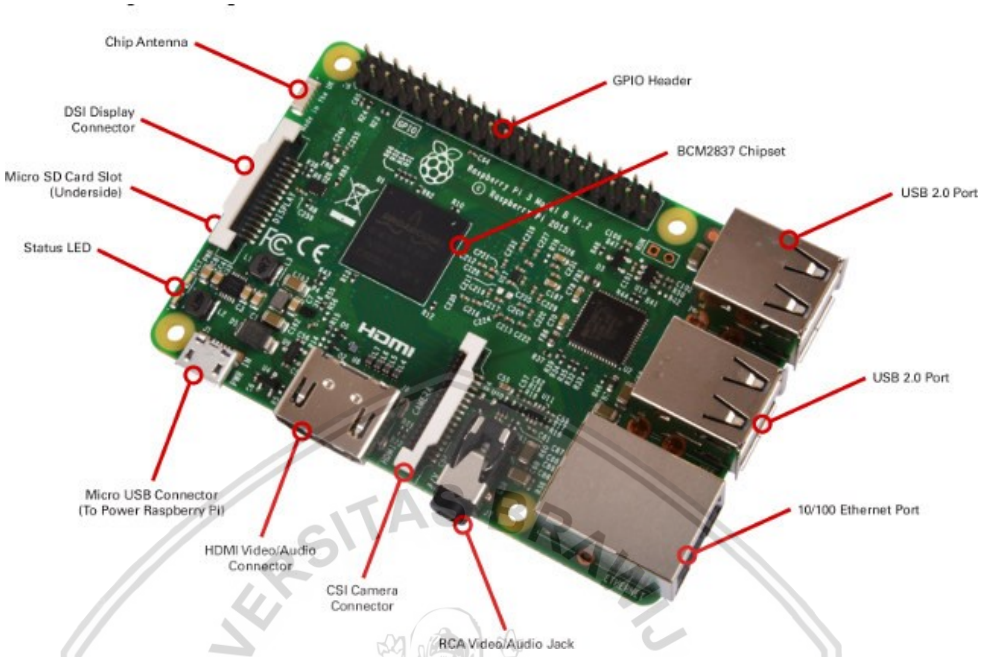
Xilinx Zynq-7010 memiliki beberapa fitur, yaitu:

- Arsitektur ARM® + FPGA yang inovatif untuk diferensiasi, analitik, dan kontrol
- OS yang luas, middleware, stack, akselerator, dan ekosistem IP
- Beberapa tingkat keamanan perangkat keras dan perangkat lunak
- Integrasi pengiriman de facto Semua platform yang Dapat Diprogram
- Kinerja tingkat sistem melalui arsitektur yang dioptimalkan
- Diarsipkan untuk menghasilkan daya sistem terendah
- Platform paling fleksibel dan skalabel untuk penggunaan ulang maksimum dan TTM terbaik
- Alat desain terkemuka di industri, C / C ++, dan abstraksi desain CL Terbuka
- Portofolio terbesar alat desain SW dan HW, SoMs, desain kit, dan desain referensi

2.2.7 Raspberry Pi Model 3

Raspberry Pi Model adalah sebuah komputer papan tunggal (single-board computer) atau SBC berukuran kartu kredit. Raspberry Pi telah dilengkapi dengan semua fungsi layaknya sebuah komputer lengkap, menggunakan SoC (*System-on-chip*) ARM yang dikemas dan diintegrasikan diatas PCB. Perangkat ini

menggunakan kartu SD untuk booting dan penyimpanan jangka panjang (Yuwono, Nugroho, & Herlyanto). Pada Gambar 2.4 adalah foto dari board Raspberry Pi model 3.



Gambar 2.4 Board Raspberry Pi

Sumber (www.cnxsoftware.com)

Raspberry Pi 3 memiliki RAM 1GB dan grafis Broadcom VideoCore IV pada frekuensi clock yang lebih tinggi dari sebelumnya yang berjalan pada 250MHz. Raspberry Pi 3 menggantikan Raspberry Pi 2 model B pada bulan Februari 2016. Kelebihannya dibandingkan dengan Raspberry Pi 2 adalah:

- 1.2GHx 64-bit quad-core ARMv8 CPU
- 802.11n Wireless LAN
- Bluetooth 4.1
- Bluetooth Low Energy (BLE)

Sama seperti Pi 2, Raspberry Pi 3 juga memiliki 4 USB port, 40 pin GPIO, *Full HDMI port*, *Port Ethernet*, *Combined 3.5mm audio jack* dan *composite video*, *Camera interface* (CSI), *Display interface* (DSI), slot kartu Micro SD (Sistem tekan-tarik, berbeda dari yang sebelum nya ditekan-tekan), dan *VideoCore IV 3D graphics core*. Raspberry Pi 3 memiliki factor bentuk identik dengan Raspberry Pi 2 dan memiliki kompatibilitas lengkap dengan Raspberry Pi 1 dan 2. Raspberry Pi 3 juga direkomendasikan untuk digunakan bagi mereka yang ingin menggunakan Pi dalam proyek-proyek yang membutuhkan daya yang sangat rendah.

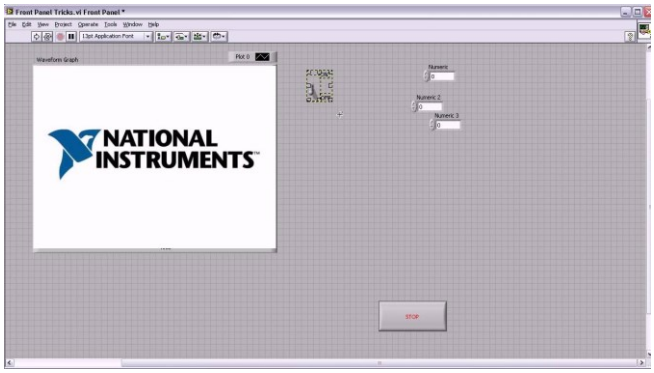
2.2.8 OpenCV

OpenCV *(Open Source Computer Vision Library*) adalah program sumber terbuka *Computer Vision* dan *machine learning*. OpenCV dibuat untuk menyediakan infrastruktur umum untuk aplikasi *Computer Vision* dan mempercepat penggunaan persepsi mesin di produk komersial. Menjadi produk berlisensi BSD, OpenCV mempermudah perusahaan untuk memanfaatkan dan memodifikasi kode. *Computer Vision* itu sendiri adalah salah satu cabang dari Bidang Ilmu Pengolahan Citra (Image Processing) yang memuingkinkan komputer dapat melihat seperti manusia (OpenCV, 2012). Dengan vision tersebut komputer dapat mengambil keputusan, melakukan aksi, dan mengenali terhadap suatu objek. Beberapa pengimplementasian dari *Computer Vision* adalah *Face Recognition, Face Detection, Face/Object Tracking, Road Tracking*, dll. OpenCV adalah sebuah API yang dikembangkan oleh perusahaan INTEL. Device yang telah menggunakan API ini salah satunya *KINECT XBOX*. Namun sayangnya XCode tidak menyediakan OpenCV Dynamic Framework. OpenCV adalah library Open Source untuk *Computer Vision* untuk C/C++, OpenCV didesain untuk aplikasi real-time, memiliki fungsi-fungsi akuisisi yang baik untuk image/video. OpenCV juga menyediakan interface ke *Integrated Performance Primitives* (IPP) Intel sehingga jika anda bisa mengoptimasi aplikasi *Vision* anda jika menggunakan prosesor Intel. OpenCV sendiri terdiri dari 5 library, yaitu :

1. CV : untuk algoritma Image processing dan Vision.
2. ML : untuk machine learning library
3. Highgui : untuk GUI, Image dan Video I/O.
4. CXCORE : untuk struktur data, support XML dan fungsi-fungsi grafis.
5. CvAux

2.2.9 NI LabVIEW

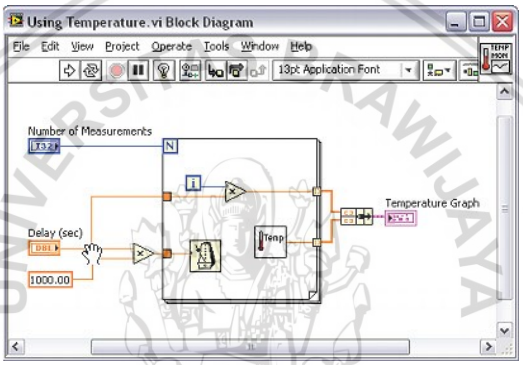
LabVIEW merupakan singkatan dari Laboratory Virtual Instrumentation Engineering Workbench yang digunakan untuk membuat program dan interface perangkat keras dan industri. LabVIEW menggunakan bahasa pemrograman berbasis grafis atau blok diagram sementara bahasa pemrograman lainnya berbasis text. Program LabVIEW dikenal dengan sebutan VI atau Virtual Instruments karena penampilan dan operasinya dapat meniru sebuah instrument. Pada LabVIEW, user pertama-tama membuat user interface atau front panel. Front panel adalah bagian dari window yang berlatar belakang abu-abu serta mengandung control dan indikator. Front panel pada Gambar 2.5 digunakan untuk membangun sebuah VI, menjalankan program dan men-debug program.



Gambar 2.5 Front Panel

Sumber (www.youtube.com)

Blok diagram pada Gambar 2.6 adalah bagian window yang berlatar belakang putih berisi sourcecode dan berfungsi sebagai instruksi untuk front panel.

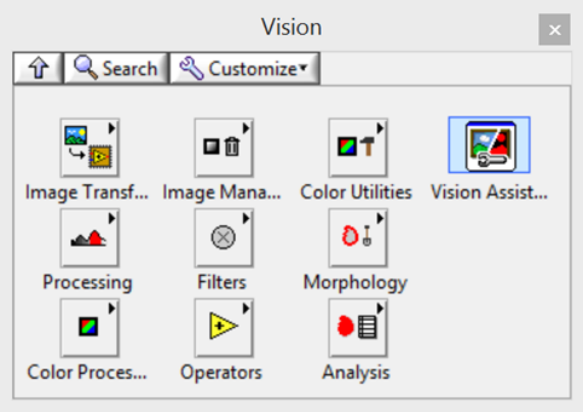


Gambar 2.5 Blok Diagram

Sumber (www.ni.com)

2.2.10 NI Vision Library

NI Vision adalah *Library* LabVIEW yang dapat mengembangkan visi computer dan aplikasi pencitraan. *NI Vision Development Module* juga mencakup fungsi pencitraan yang sama untuk *LabWindows / CVI* dan lingkungan pengembangan C lainnya; serta kontrol *ActiveX* untuk *Visual Basic. Vision Assistant* dan produk *Vision Development Module* adalah perangkat lunak yang memungkinkan untuk membuat program dengan cepat tanpa harus melakukan pemograman apapun. Pada Gambar 2.7 dapat dilihat tampilan dari *NI Vision Library* pada program *LabVIEW.*

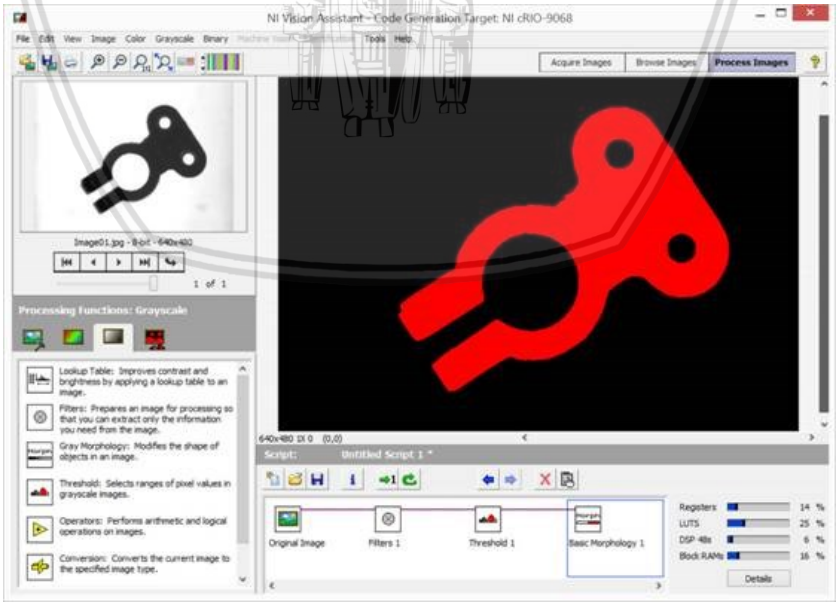


Gambar 2.7 Tampilan *Library NI VIsion*

Sumber (www.ni.com)

2.2.11 NI Vision Assistant

NI Vision Asisstant adalah alat untuk *prototyping* dan pengujian aplikasi pengolah Gambar. Kita bisa membuat algoritma *custom* dengan fitur *Vision Assistant scripting*, yang mencatat setiap langkah algoritma pengolahan Anda. *NI Vision Asisstant* juga bisa menghasilkan LabVIEW VI yang bekerja berdasarkan skrip yang dibuat atau menghasilkan file pembangun yang berisi panggilan yang dibutuhkan untuk menjalankan skrip di C atau Visual Basic. *NI Vision Asisstant* disertakan sebagai bagian dari *Vision Development Module.* Gambar 2.8 adalah tampilan jendela *NI Vision Assistant*



Gambar 2.8 Tampilan *NI Vision Assistant*

Sumber (www.ni.com)

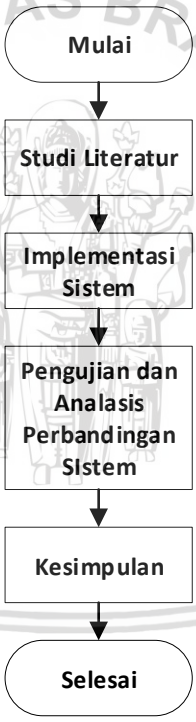
Vision Assistant menggunakan *Library NI Vision* namun dapat digunakan secara independen dari lingkungan pengembangan lainnya. Selain menjadi alat untuk membuat prototip sistem *Vision*, Kita bisa menggunakan *Vision Assistant* untuk mengetahui bagaimana fungsi pengerjaan Gambar yang berbeda. Antarmuka *Vision Assistant* membuat prototip aplikasi Anda lebih mudah dan efisien karena fitur seperti jendela referensi yang menampilkan Gambar asli Anda, jendela skrip yang menyimpan langkah pengolahan Gambar Anda, dan jendela pengolahan yang mencerminkan perubahan pada Gambar Anda saat Anda menerapkan parameter baru

BAB 3 METODOLOGI

Bab ini menjelaskan metode yang digunakan untuk melakukan penelitian. Tipe penelitian ini adalah Anailisis yaitu bersifat observasi perbandingan pengolahan antara dua buah sistem

3.1 Alur Metode Penelitian

Bagian metode penelitian adalah penjelasan mengenai langkah-langkah yang dilakukan untuk menyusun dan menyelesaikan penelitian yag dimulai dari, studi literatur, implementasi sistem, pengujian dan analisis sistem terhadapa sistem dari penelitian yang dibuat. Untuk kesimpulan dan saran akan disertakan agar dapat memberikan Gambaran untuk melakukan pengembangan sistem selanjutnya. Langkah-langkah dalam alur pengerjaan penelitian ini ditunjukan pada Gambar 3.1.



Gambar 3.1 Diagram Alir Penelitian

3.2 Studi Literatur

Studi literatur dibutuhkan untuk mencari informasi dan referensi yang dilakukan dari penelitian-penelitian sebelumnya. Dari proses studi literature kemudian diperoleh dasar-dasar teori yang akan digunakan untuk penelitian ini. Beberapa dasar teori yang digunakan dalam penelitian ini diantaranya Metode *Grayscale, Gaussian Filter, Laplacian Edge, Sobel Edge, FPGA Ni myRIO*, Aplikasi LabVIEW *myRIO, Raspberry Pi, OpenCV, NI Vision Library, NI Vision Development Module, NI Vision Assistant dan Xilinx Zynq-7010*

3.3 Perancangan Sistem

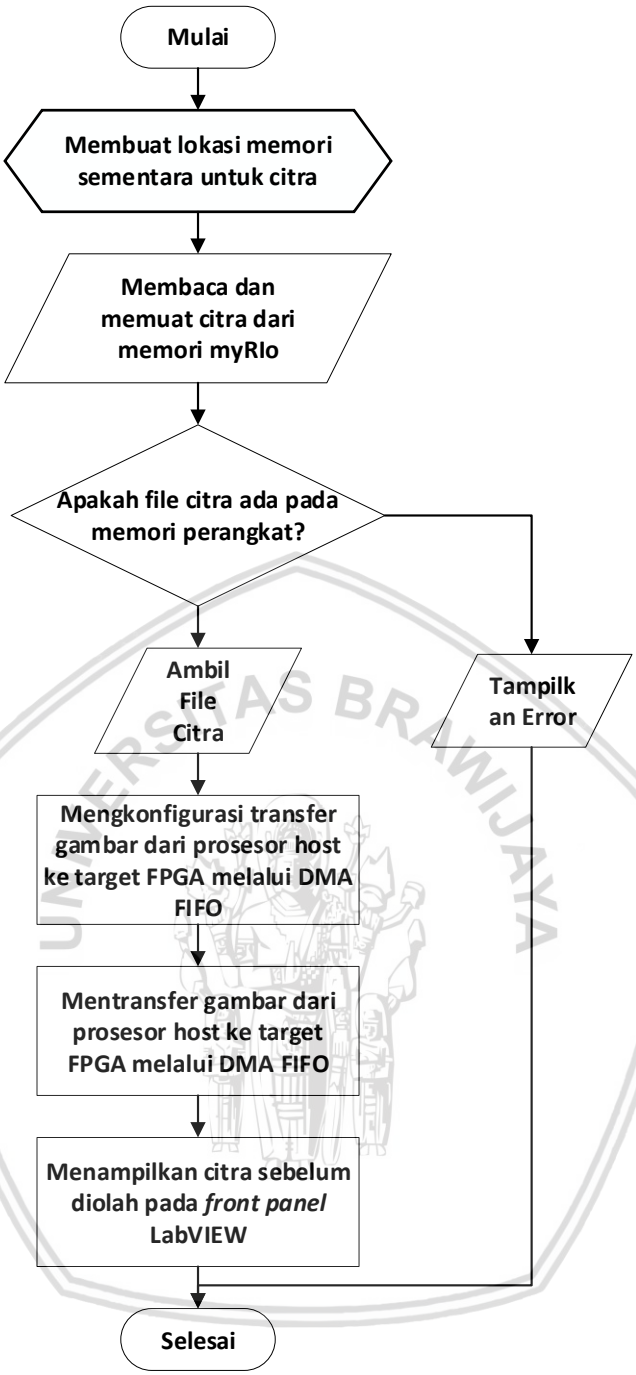
Perancangan sistem merupakan tahap yang dilakukan setelah semua kebutuhan sistem dianalisis dan terpenuhi. Pada penelitian ini, dibutuhkan perancangan perangkat lunak untuk merancanga algoritma pengolahan citra. Sedangkan pada penelitian ini tidak dibutuhakn perancangan perangkat keras karena sistem pada kedua *platform* bekerja secara independen dan tidak membutuhkan komponen tambahan

3.3.1 Perancangan pada platform FPGA

Pada perancangan pada *platform* FPGA akan dijelaskan cara sistem bekerja, berupa algoritma dan metode yang akan digunakan pada sistem

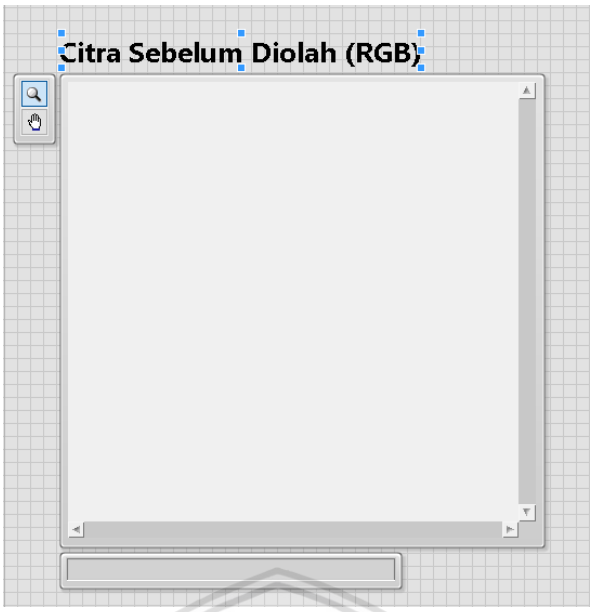
3.3.1.1 Fungsi Mengambil dan Menampilkan Citra Sebelum Diolah

Pada fungsi ini sistem dapat mengambil file citra dari memori perangkat yang akan melakukan pengolahan citra. Selain dapat mengambil file citra, sistem juga dapat menampilkan citra yang akan diolah dan dapat dilihat oleh *user*. Untuk merancang fungsi ini, penulis menggunakan beberapa VI dari LabVIEW, VI yang pertama adalah VI *IMAQ Create* yang berfungsi membuat lokasi memori sementara untuk citra. Selanjutnya adalah VI *IMAQ ReadFile* yang berfungsi membaca dan memuat citra dari memory myRIO, VI *IMAQ FPGA Configure Image Transfer to Target* yang berfungsi mengkonfigurasi transfer gambar dari prosesor host ker target FPGA melalui DMA FIFO dan yang terakhir adalah VI I*MAQ FPGA Image Transfer to Target* yang berfungsi mentrasnfer gambar dari prosesor host ke target FPGA melalui DMA FIFO. Setelah semua VI ini tekoneksi dengan benar, maka citra sebelum diolah dapat ditampilkan pada *front panel* dari aplikasi LabVIEW. Pada gambar 3.2 adalah flowchart cara kerja dari fungsi mengambil dan menampilakan citra sebelum diolah.



Gambar 3.2 Flowchart Mengambil dan Menampilkan Citra Sebelum Diolah

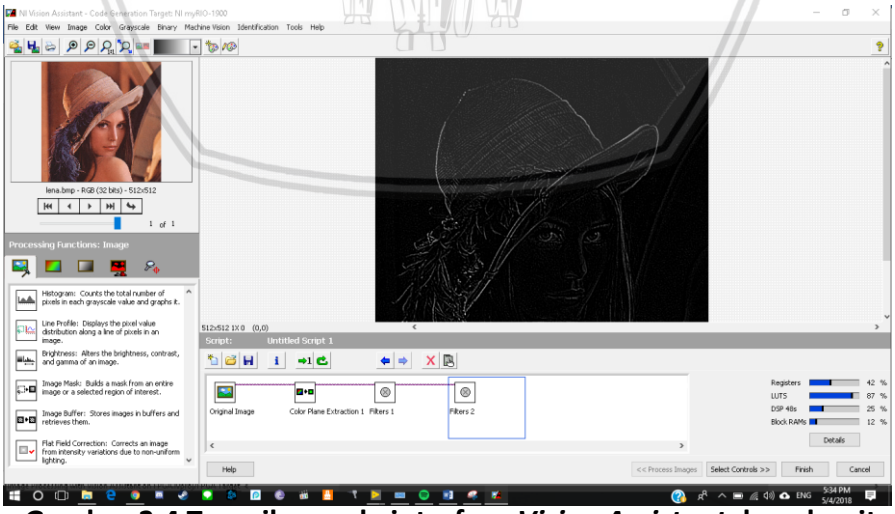
Pada gambar 3.3 adalah gambar dari tampilan front panel LabVIEW dimana citra aka ditampilkan sebelum diolah



Gambar 3.3 *Interface Front Panel* Menampilkan Citra Sebelum Diolah pada LabVIEW

3.3.1.2 Fungsi Mengolah Citra

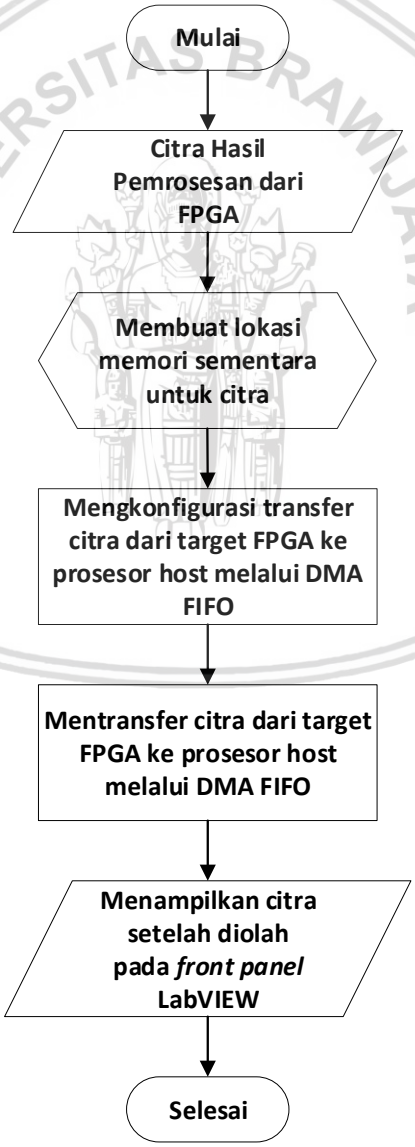
Pada peneilitian ini, penulis menggunakan fungsi *Color Plane Extraction* yang berfungsi mengekstrak salah satu dari tiga bidang warna *(Red, Green, Blue, Hue, Saturation, Luminance, Value, Intensity)* dari sebuah citra. Disini penulis *Luminance Plane* untuk mengolah citra RGB menjadi *grayscale*. Setelah citra diolah menjadi *grayscale*, citra akan di *smoothing* dengan fungsi *Filters* di *Vision Assistant* dengan algoritma *Gaussian Blur* agar citra lebih halus dan lebih mudah untuk mendeteksi tepi. Selanjutnya citra akan diolah dengan algoritma *Sobel Edge* atau *Laplacian Edge* dan terdeteksi tepinya. Pada gambar 3.4 adalah tampilan pada interface *Vision Assistant* dan algoritma yang digunakan



Gambar 3.4 Tampilan pada interface *Vision Assistant* dan algoritmanya

3.3.1.3 Fungsi Mengambil dan Menampilkan Citra Setelah Diolah

Pada fungsi ini sistem dapat mengambil file citra dari memori perangkat yang telah melakukan pengolahan citra. Selain dapat mengambil file citra, sistem juga dapat menampilkan citra yang telah diolah dan dapat dilihat oleh *user.* Untuk merancang fungsi ini, penulis menggunakan beberapa VI dari LabVIEW, VI yang pertama adalah VI *IMAQ Create* yang berfungsi membuat lokasi memori sementara untuk citra. Selanjutnya adalah VI *IMAQ FPGA Configure Image Transfer from Target* yang berfungsi mengkonfigurasi transfer citra dari FPGA ke prosesor host melalu DMA FIFO dan yang terakhir adalah VI I*MAQ FPGA Image Transfer from Target* yang berfungsi mentransfer citra dari FPGA ke prosesor host melalu DMA FIFO. Setelah semua VI ini tekoneksi dengan benar, maka citra sebelum diolah dapat ditampilkan pada *front panel* dari aplikasi LabVIEW. Pada Gambar 3.5 adalah flowchart cara kerja dari fungsi mengambil dan menampilakan citra setelah diolah.

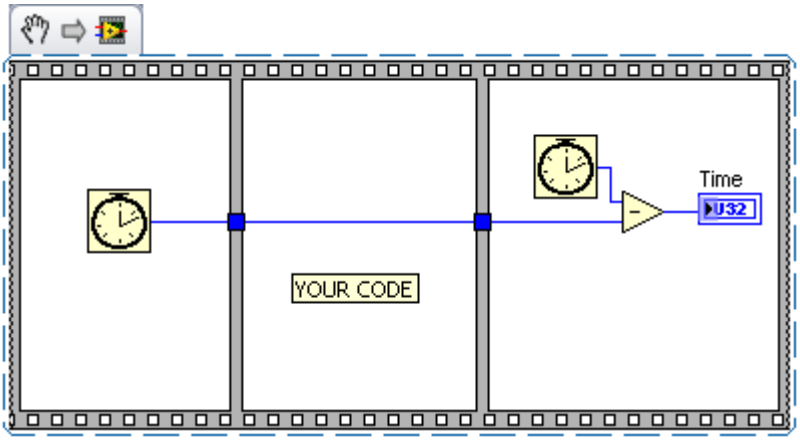


Gambar 3.5 Flowchart mengambil dan menampilkan citra setelah diolah

Gambar 3.6 *Interface Front Panel* Menampilkan Citra Setelah Diolah pada LabVIEW

3.3.1.4 Fungsi Menghitung Waktu Pengolahan Citra

Pada fungsi ini sistem dapat mengambil data waktu pengolahan citra saat sistem bekerja. Waktu dihitung dari proses mengambil dan menampilkan citra sebelum diolah, proses pengolahan citra dan proses mengambil dan menampilkan citra setelah diolah. Untuk menghitung waktu pengolahan citra pada program LabVIEW, digunakan *Flat Sequence Structure* yang berfungsi untuk memastikan subdiagram berfungsi secara berurutan dari diagram kiri ke diagram kanan. Setelah *Flat Sequence Structure* terbentuk, digunakan fungsi *Tick Count* yang berfungsi sebagai *timer dan* menghitung nilai waktu dalam satuan *millisecond.* Pada gambar 3.7 adalah contoh fungsi *Tick Count* didalam *Flat Sequence Structure* yang digunakan pada program LabVIEW.



Gambar 3.7 Contoh Fungsi *Tick Count* didalam *Flat Sequence Structure*

3.3.2 Perancangan pada Platform Mikrokomputer

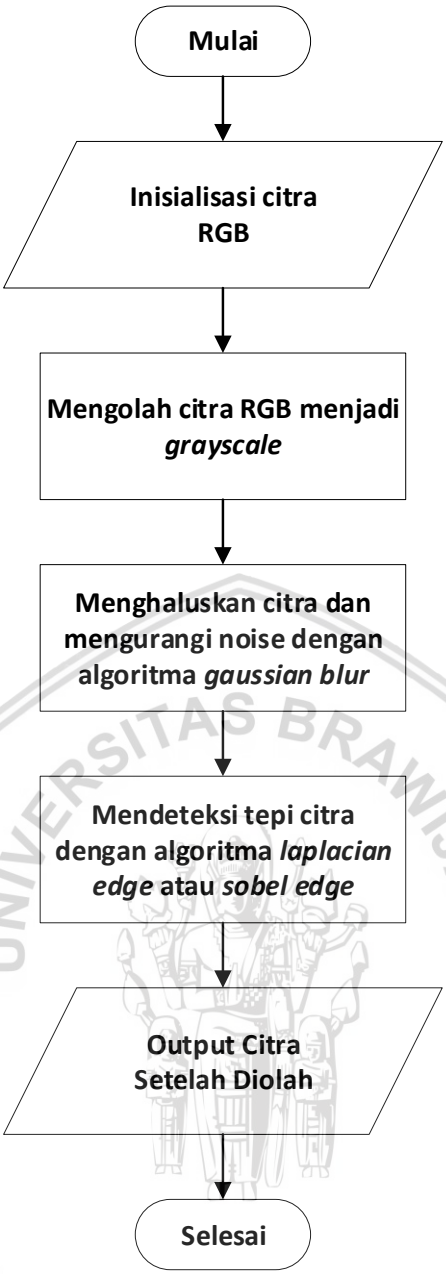
Pada perancangan dan implementasi ada *platform* Mikrokomputer akan dijelaskan cara sistem bekerja, berupa algoritma dan metode yang akan digunakan pada sistem mikrokomputer di Raspberry Pi

3.3.2.1 Fungsi Mengambil dan Menampilkan Citra Sebelum Diolah

Pada fungsi ini sistem dapat mengambil file citra dari memori perangkat yang akan melakukan pengolahan citra. Selain dapat mengambil file citra, sistem juga dapat menampilkan citra yang akan diolah dan dapat dilihat oleh *user*. Untuk mengambil citra, pertama inisialisasi dulu variablenya dan menggunakan fungsi *cv2.imread()* yang tersedia dan kita deklarasikan nama file untuk citra yang akan diolah. File citra yang akan diolah diletakkan satu folder dengan kode program agar memudahkan untuk eksekusi. Untuk menampilkan citra sebelum diolah, digunakan fungsi *cv2.imshow()* yang tersedia pada Library OpenCV.

3.3.2.2 Fungsi Mengolah Citra

Fungsi mengolah citra merupakan fungsi utama pada sistem yang berfungsi untuk mengubah citra RGB yang telah di input user menjadi citra *Gaussian Filter, Laplacian Edge* dan *Sobel Edge.* Pertama citra RGB diolah menjadi citra *grayscale* dengan fungsi *cv2.COLOR_RGB2GRAY* yang ada di Library OpenCV. Setelah citra menjadi *grayscale*, citra akan dihaluskan dan dikurangi noisenya dengan fungsi *cv2.GaussianBlur()* supaya deteksi tepi menjadi lebih akurat. Lalu citra akan dideteksi tepinya dengan algoritma *laplacian edge* atau *sobel edge* dan terdeteksi tepinya. Berikut adalah alur kerja fungsi mengolah citra yang dapat dilihat pada Gambar 3.8 berikut:



Gambar 3.8 Flowchart Fungsi Mengolah Citra

3.3.2.3 Fungsi Mengambil dan Menampilkan Citra Sebelum Diolah

Pada fungsi ini sistem dapat mengambil file citra dari memori perangkat yang telah melakukan pengolahan citra. Selain dapat mengambil file citra, sistem juga dapat menampilkan citra yang telah diolah dan dapat dilihat oleh *user.* Untuk mengambil citra yang telah diolah, digunakan fungsi *cv2.imwrite()* untuk menulis citra yang telah kita olah ke memori Raspberry Pi. Sedangkan untuk menampilkan citra setelah diolah, digunakan fungsi *cv2.imshow()* yang tersedia pada Library OpenCV.

3.3.2.4 Fungsi Menghitung Waktu Pengolahan Citra

Pada fungsi ini sistem dapat mengambil data waktu pengolahan citra saat sistem bekerja. Waktu dihitung dari proses mengambil dan menampilkan citra sebelum diolah, proses pengolahan citra dan proses mengambil dan menampilkan citra setelah diolah. Untuk menghitung waktu menggunakan *Library* OpenCV, digunakan fungsi yang telah tersedia pada OpenCV yaitu *cv2.getTickCount()* dan *cv2.getTickFrequency().* Fungsi *cv2.getTickCount()* digunakan di awal dan akhir kode program sebagai penanda awal waktu awal dan akhir program di eksekusi. Lalu di akhir program waktu akhir dikurangi waktu awal dan dibagi oleh fungsi *cv2.getTickFrequency()* sehingga dapat dihasilkan waktu eksekusi kode program

3.4 Implementasi

Implementasi merupakan tahapan untuk merealisasikan sistem yang telah dirancang. Implementasi perancangan dilakukan setelah analisis kebutuhan sistem selesai dilakukan. Pada tahap ini dilakukan pengimplementasian semua gagasan dan ide baik desain maupun perhitungan menjadi sebuah satu kesatuan. Implementasi dibagi menjadi dua bagian yaitu implementasi perangkat keras dan implementasi perangkat lunak.

3.4.1 Implementasi Perangkat Keras

Tahap ini implementasi dilakukan pada perangkat keras yang telah dianalisis. Implementasi perangkat keras pada FPGA, sistem dihubungkan dengan *power supply* dan dihubngkan dengan kabel USB ke PC. Implementasi perangkat keras pada Raspberry Pi Sistem dihubungkan dengan *power supply* dan menggunakan kabel Ethernet ke PC untuk remote Raspberry secara local

3.4.2 Implementasi Perangkat Lunak

Tahap ini implementasi dilakukan pada perangkat lunak yang telah dianalisis. Implementasi perangkat lunak pada FPGA menggunakan aplikasi *NI LabVIEW myRIO*. Pada *NI LabVIEW myRIO* program-program sistem dibuat. Program-program yang dibuat antara lain programakuisisi citra dari host ke FPGA, program pengolahan citra, dan program akuisisi citra dari FPGA ke host lalu bisa ditampilkan ke *Front Panel.* Implementasi perangkat lunak pada Raspberry Pi menggunakan Terminal pada *OS Raspbian* dan *OpenCV Library* . Pada Terminal program-program sistem dibuat. Program yang dibuat antara lain membaca file citra pada memory Raspberry Pi, program pengolahan citra dan program membuat file hasil pengolahan citra.

3.5 Metode Pengumpulan Data

Metode pengumpulan data pada penelitian ini mengambil data waktu pemrosesan pengolahan citra dari kedua sistem yaitu FPGA dan Mikrokomputer, data waktu diambil dari awal sistem bekerja yaitu proses mengambil dan menampilkan citra sebelum diolah, proses mengolah citra dan proses mengambil

dan menampilkan citra setelah diolah. Dalam pengumpulan data, ada beberapa skenario yang akan dilakukan untuk pengujian sistem, yaitu:

1. Pengujian terhadap file image dengan resolusi yang berbeda
2. Penguian terhadap algoritma pengolahan citra yang berbeda
3. Pengujian terhadap waktu pengolahan citra antara FPGA dan Raspberry Pi

3.6 Metode Analisis Hasil

Pada tahap ini akan dilakukan analisis perbandingan kedua sistem hasil pengujian yang mana akan dilaksanakan sesuai dengan parameter perancangan sistem dengan membandingkan performa sistem berdasarkan waktu pengolahan citra.

3.7 Kesimpulan

Kesimpulan merupakan Gambaran dari hasil ,implementasi, pengujian dan analisis perbandingan antara kedua sistem. Kesimpulan disusun berdasarkan hasil pengujian dan analisis yang dibuat. Isi dari kesimpulan diharapkan dapat menjadi acuan pada penelitian lain untuk pengembangan sistem yang lebih baik. Di dalam penulisan akhir bertujuan untuk memberikan kemudahan kepada peneliti selanjutnya apabila ingin melanjutkan penelitian ini.

BAB 4 IMPLEMENTASI

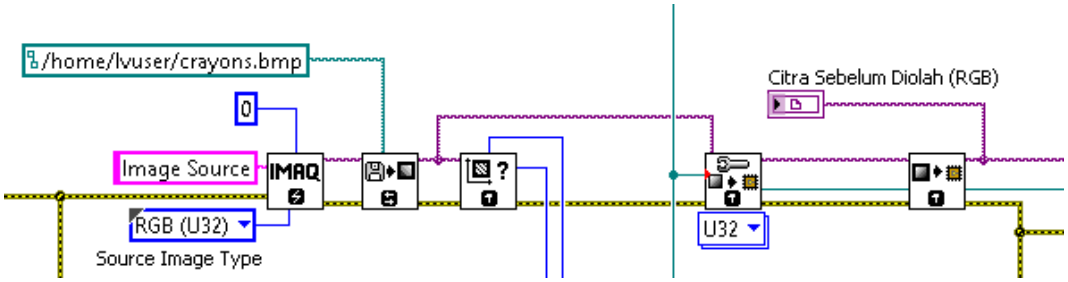
Bab ini adalan implementasi dari kedua sistem yaitu pada FPGA dan Mikrokomputer sehingga sistem dapat bekerja dengan baik sesuai tujuan yang ingin dicapai

4.1 Implementasi pada FPGA

Implementasi sistem pada FPGA dilakukan pada aplikasi LabVIEW dan menggunakan library day *Vision Assistant* sehingga sistem bisa dilakukan pengujian dan analisis.

4.1.1 Fungsi Mengambil dan Menampilkan Citra Sebelum Diolah

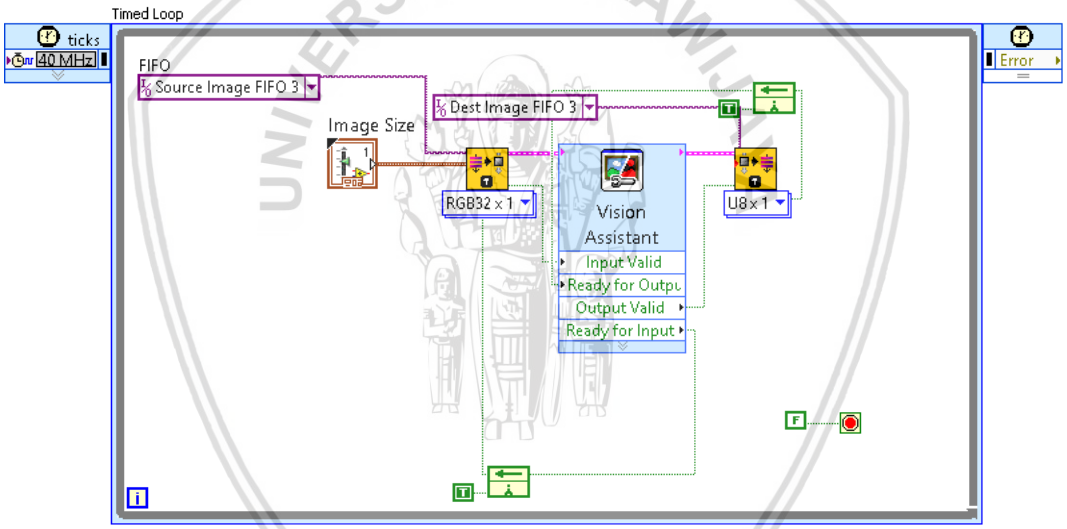
Pada fungsi ini sistem dapat mengambil file citra dari memori perangkat yang akan melakukan pengolahan citra. Selain dapat mengambil file citra, sistem juga dapat menampilkan citra yang akan diolah dan dapat dilihat oleh *user.* Untuk mengambil data citra, diinisialisasikan memory penyimpanan sementara untuk menyimpan citra dengan bantuan VI *IMAQ Create* dengan *Image Type* adalah RGB (U32) sesuai dengan citra awal yang akan diolah. Setelah membuat memory penyimpanan untuk citra yang akan diolah, diperlukan perintah untuk membaca direktori file citra yang akan diolah. Ini bisa terjadi dengan bantuan VI *IMAQ ReadFIle.* VI ini berfungsi untuk membaca file yang ada di terdapat di memory myRIO dan memuatnya ke memory sementara yang telah diinisalisasi tadi. Selanjutnya dibutuhkan VI *IMAQ FPGA Configure Image Transfer to Target.* VI ini berfungsi untuk mengkonfigurasi transfer Gambar dari prosesor host ke target FPGA melalui DMA FIFO. Dan yang terakhir adalah VI *IMAQ FPGA Image Transfer to Target* yang berfungsi mentransfer Gambar dari prosesor host ke target FPGA melalui DMA FIFO dan lalu ditampilkan citra sebelum diolah pada *front panel* LabVIEW. Pada Gambar 4.1 terdapat Gambar implementasi pengambilan dan menampilkan citra sebelum diolah pada LabVIEW. Pada Gambar tersebut menampilkan VI apa saja yang digunakan dan bagaimana koneksi antar VI tersebut.



Gambar 4.1 Blok Diagram Mengambil dan Menampilkan Citra Sebelum Diolah pada LabVIEW

4.1.2 Fungsi Mengolah Citra

Fungsi Mengolah Citra merupakan fungsi utama pada sistem yang berfungsi untuk mengubah citra RGB yang telah di input user menjadi citra yang telah diolah dengan algoritma *Gaussian Filter, Laplacian Edge* atau *Sobel Edge.* Dengan bantuan NI *Vision Assistant*, penulis dapat menerapkan algortima-algoritma pengolahan citra dengan cepat tanpa harus merancang algoritma tersebut. Sebelum citra diproses oleh *Vision Assistant,* sistem harus mengambil citra dari FIFO FPGA menggunakan VI *IMAQ FPGA FIFO to Pixel Bus.* VI ini berfungsi untuk membaca piksel Gambar dari FIFO yang ditentukan dan mengembalikan kluster Pixel Bus dari jenis yang diminta. Setelah citra diambil, lalu citra akan diproses pada VI *Vision Assistant. Vision Assistant* bisa menggunakan algoritma yang disediakan oleh *Vision Assistant* untuk mengolah citra. Setelah citra diolah , tipe citra yang tadinya adalah RGB 32-bit menjadi U 8-bit. Citra yang telah diolah tadi dikirim kembali pixel bus pada FIFO FPGA yang telah terdefinisi menggunakan VI *IMAQ FPGA Pixel Bus to FIFO VI.* Pada Gambar 4.2 terdapat blok diagram untuk fungsi mengolah citra dan koneksi tiap VI didalamnya

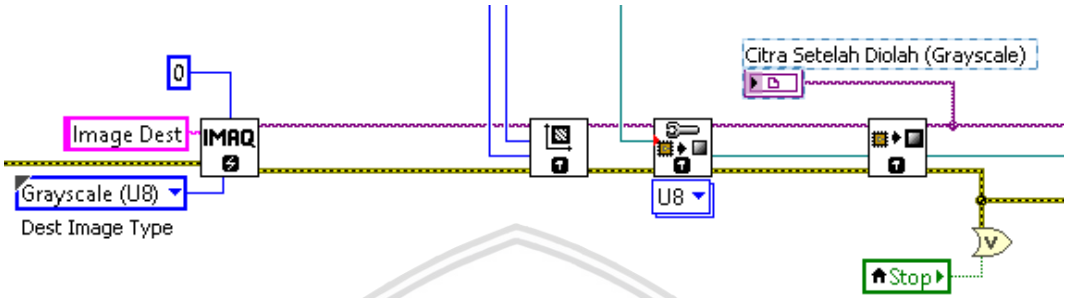


Gambar 4.2 Blok Diagram Fungsi Mengolah Citra

4.1.3 Fungsi Mengambil dan Menampilkan Citra Setelah Diolah

Pada fungsi ini sistem dapat mengambil file citra dari memori perangkat yang telah melakukan pengolahan citra. Selain dapat mengambil file citra, sistem juga dapat menampilkan citra yang telah diolah dan dapat dilihat oleh *user.* Setelah citra berhasil diolah, sistem membutuhkan penyimpanan sementara untuk menyimpan citra dengan bantuan VI *IMAQ Create* dengan *Image Type* adalah Grayscale (U8) sesuai dengan tipe citra yang telah diolah. Kemudian, sistem membutuhkan konfigurasi menggunakan VI *IMAQ FPGA Configure Image Transfer from Target* yang berfungsi mengkonfigurasi transfer Gambar dari target FPGA ke prosesor host melalui DMA FIFO. Dan yang terakhir ini sistem mentransfer citra yang telah diolah dari target FPGA ke prosesor host melalui DMA FIFO dengan

bantuan VI *IMAQ FPGA Image Transfer from Target*. Selain dapat mengambil file citra, fungsi ini juga dapat menampilkan citra yang telah diolah dan dapat dilihat oleh *user* pada *front panel* LabVIEW. Pada Gambar 4.3 terdapat Gambar implementasi pengambilan dan menampilkan citra setelah diolah pada LabVIEW. Pada Gambar tersebut menampilkan VI apa saja yang digunakan dan bagaimana koneksi antar VI tersebut

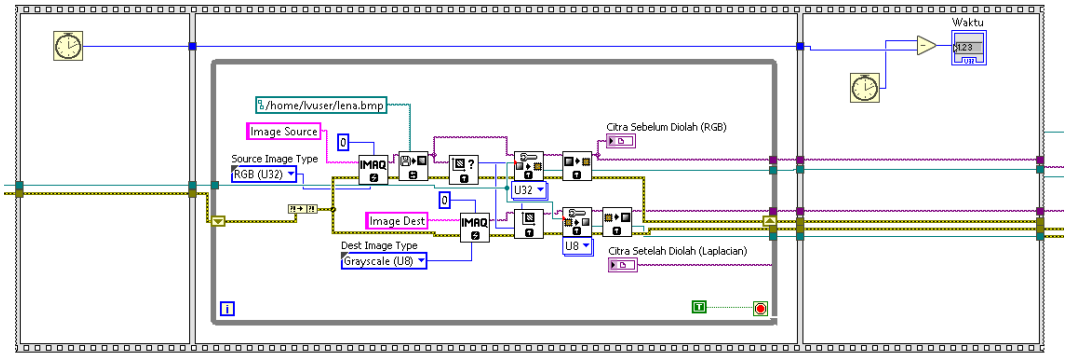


Gambar 4.3 Blok Diagram Mengambil dan Menampilkan Citra Sebelum Diolah pada LabVIEW

4.1.4 Fungsi Menghitung Waktu Pengolahan Citra

Pada fungsi ini sistem dapat mengambil data waktu pengolahan citra saat sistem bekerja. Waktu dihitung dari proses mengambil dan menampilkan citra sebelum diolah, proses pengolahan citra dan proses mengambil dan menampilkan citra setelah diolah. Untuk menghitung waktu pengolahan citra pada program LabVIEW, digunakan *Flat Sequence Structure* yang berfungsi untuk memastikan subdiagram berfungsi secara berurutan dari diagram kiri ke diagram kanan. Setelah *Flat Sequence Structure* terbentuk, digunakan fungsi *Tick Count* yang berfungsi sebagai *timer dan* menghitung nilai waktu dalam satuan *millisecond.*

Pada diagram paling kiri, ada fungsi *Tick Count* yang tereksekusi saat pertama kali saat program berjalan. Selanjutnya *Wire* dari *Tick Count* tadi melewati diagram tengah pada program. Diagram tengah ini adalah tempat dari proses mengambil dan menampilkan citra sebelum diolah, proses pengolahan citra dan proses mengambil dan menampilkan citra setelah diolah bekerja. Lalu *Wire* dari *Tick Count* sampai di diagram paling kanan, pada diagram ini ada satu fungsi *Tick Count* lagi sebagai acuan waktu berakhirnya program bekerja. Lalu *Tick Count* pada diagram paling kanan dikurangi *Tick Count* diagram paling kiri dan didapatkanlah waktu pengolahan citra. Setelah didapat waktunya, nilai waktu ditampilkan pada *front panel* dalam satuan *millisecond.* Pada Gambar 4.4 adalah Gambar blok diagram penggunan *Flat Sequence Structure* pada program LabVIEW



Gambar 4.4 Penggunaan *Flat Sequence Structure* untuk menghitung waktu pada LabVIEW

4.2 Implementasi pada Mikrokomputer

Implementasi sistem pada Mikrokomputer dilakukan pada OS Raspbian dan menggunakan library day *OpenCV* sehingga sistem bisa dilakukan pengujian dan analisis.

4.2.1 Fungsi Mengambil dan Menampilkan Citra Sebelum Diolah

Pada fungsi ini sistem dapat mengambil file citra dari memori perangkat yang akan melakukan pengolahan citra. Selain dapat mengambil file citra, sistem juga dapat menampilkan citra yang akan diolah dan dapat dilihat oleh *user*. Untuk mengambil citra, pertama inisialisasi dulu variablenya dan menggunakan fungsi *cv2.imread()* yang tersedia dan kita deklarasikan nama file untuk citra yang akan diolah. Pada Tabel 4.1 merupakan contoh dari mengambilkan citra yang telah disimpan pada memori Raspberry Pi dan digunakan dengan library dari OpenCV

Tabel 4.1 Fungsi mengambil citra

```
1 image = cv2.imread('namafile.bmp')
```

Untuk menampilkan citra sebelum diolah, digunakan fungsi *cv2.imshow()* yang tersedia pada Library OpenCV. Pada Tabel 4.2 adalah contoh dari penggunaan fungsi *cv2.imshow()* untuk menampilkan citra

Tabel 4.2 Fungsi Menampilkan Citra Sebelum Diolah

```
1 cv2.imshow('color_image',image)
```

4.2.2 Fungsi Mengolah Citra

Fungsi mengolah citra merupakan fungsi utama pada sistem yang berfungsi untuk mengubah citra RGB yang telah di input user menjadi citra *Gaussian Blur, Laplacian Edge* dan *Sobel Edge.* Pada Library OpenCV, telah disediakan fungsi *cv2.cvtColor()* yang bisa merubah citra menjadi *color space* yang berbeda. Karena penelitian ini membutuhkan citra RGB menjadi *Grayscale,* dipilihlah parameter

cv2.COLOR_RGB2GRAY. Pada Tabel 4.3 adalah contoh dari penggunaan fungsi *cv2.cvtColor()* untuk mengolah citra

Tabel 4.3 Mengolah citra menjadi grayscale

```
1 gray_image = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY)
```

Setelah citra RGB diolah menadi citra *grayscale,* selanutnya citra tersebut harus diolah kembali dengan algoritma *Gaussian Blur* yang berfungsi agar citra lebih halus dan lebih mudah untuk mendeteksi tepi. Untuk menerapkan algortima *Gaussian Blur,* OpenCV telah menyediakan fungsi tersebut dengan bantuin fungsi *cv2.GaussianBlur().* Pada Tabel 4.4 adalah contoh dari penggunaan fungsi *cv2.GaussianBlur()*

Tabel 4.4 Mengolah citra dengan algortima *Gaussian Blur*

```
1 denoised = cv2.GaussianBlur(gray,(3,3),0)
```

Proses selanjutnya setelah citra diolah dengan algoritma *Gaussian Blur* adalah deteksi tepi. Pada penelitian ini digunakan dua algoritman deteksi tepi, yaitu *Laplacian Edge* dan *Sobel Edge.* Untuk mengolah citra dengan algoritma deteksi tepi*,* OpenCV telah menyediakan library untuk mendeteksi tepi dengan algoritma *Laplacian Edge* dan *Sobel Edge.* Untuk algoritma *Laplacian Edge* digunakan fungsi *cv2.Laplacian()* sedangkan untuk algoritma *Sobel Edge* digunakan fungsi *cv2.Sobel()* yang tersedia pada Library OpenCV. Pada Tabel 4.5 dan Tabel 4.6 adalah contoh penggunaan dari fungsi *cv2.Laplacian()* dan *cv2.Sobel()* pada OpenCV

Tabel 4.5 Mengolah citra dengan algortima *Laplacian Edge*

```
1 filter = cv2.Laplacian(denoised,cv2.CV_8U)
```

Tabel 4.6 Mengolah citra dengan algortima *Sobel Edge*

```
1 sobelx = cv2.Sobel(denoised,cv2.CV_8U,1,0,ksize=3)
2 sobely = cv2.Sobel(denoised,cv2.CV_8U,0,1,ksize=3)
3 sobel = sobelx + sobely
```

4.2.3 Fungsi Mengambil dan Menampilkan Citra Setelah Diolah

Pada fungsi ini sistem dapat mengambil file citra dari memori perangkat yang telah melakukan pengolahan citra. Selain dapat mengambil file citra, sistem juga dapat menampilkan citra yang telah diolah dan dapat dilihat oleh *user.* Untuk mengambil citra yang telah diolah, digunakan fungsi *cv2.imwrite()* untuk menulis citra yang telah kita olah ke memori Raspberry Pi. Pada Tabel 4.7 adalah contoh penggunaan fungsi *cv2.imwrite()* untuk mengambil citra

Tabel 4.7 Fungsi Mengambil Citra

```
1 cv2.imwrite('namacitra.jpg',laplacian)
```

Untuk menampilkan citra setelah diolah, digunakan fungsi *cv2.imshow()* yang tersedia pada Library OpenCV. Pada Tabel 4.8 adalah contoh dari penggunaan fungsi *cv2.imshow* untuk menampilkan citra

Tabel 4.8 Fungsi Menampilkan Citra

```
1  cv2.imshow('Laplacian Edge',laplacian)
```

4.2.4 Fungsi Menghitung Waktu Pengolahan Citra

Pada fungsi ini sistem dapat mengambil data waktu pengolahan citra saat sistem bekerja. Waktu dihitung dari proses mengambil dan menampilkan citra sebelum diolah, proses pengolahan citra dan proses mengambil dan menampilkan citra setelah diolah. Untuk menghitung waktu menggunakan *Library* OpenCV, digunakan fungsi yang telah tersedia pada OpenCV yaitu *cv2.getTickCount()* dan *cv2.getTickFrequency().*

Fungsi *cv2.getTickCount()* berfungsi mengembalikan jumlah *clock-cycles* setelah kejadian referensi (seperti saat mesin dinyalakan) ke saat fungsi ini dipanggil. Jadi jika fungsi ini dipanggil sebelum dan sesudah eksekusi fungsi, bisa didapatkan jumlah *clock-cycles* yang digunakan untuk menjalankan program ini. Fungsi *cv2.getTickFrequency()* berfungsi mengembalikan *frekuensi clock-cycle,* atau jumlah *clock-cycle* per detik.Nilai waktu pengolahan citra akan muncul pada jendela Terminal. Jadi untuk menemukan waktu eksekusi dalam hitungan detik, bisa dilihat pada Tabel 4.9 penggunaan fungsi yaitu *cv2.getTickCount()* dan *cv2.getTickFrequency()* untuk menghitung waktu pengolahan citra:

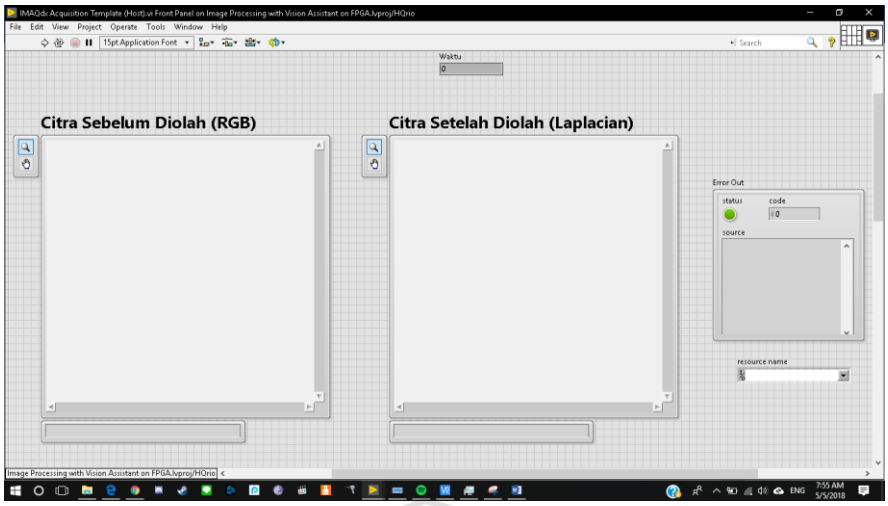
Tabel 4.9 Fungsi Menghitung Waktu Pengolahan Citra

```
1  e1 = cv2.getTickCount()
2  img = cv2.imread('lena.bmp')
3  gray = cv2.cvtColor(img, cv2.COLOR_RGB2GRAY)
4  denoised = cv2.GaussianBlur(gray,(3,3),0)
5  cv2.imshow('Original',img)
6  cv2.imshow('Gaussian Blur',denoised)
7  e2 = cv2.getTickCount()
8  time = (e2 - e1)/ cv2.getTickFrequency()
9  print ( time )
```

4.3 Implementasi Eksekusi Sistem pada FPGA

Pada tahap implementasi pengolah citra pada FPGA akan dijelaskan tahap-tahap yang dilakukan agar sistem dapat bekerja sesuai dengan yang diharapkan. Implementasi pengolah citra pada FPGA dilakukan dengan tahapan-tahapan sebagai berikut :

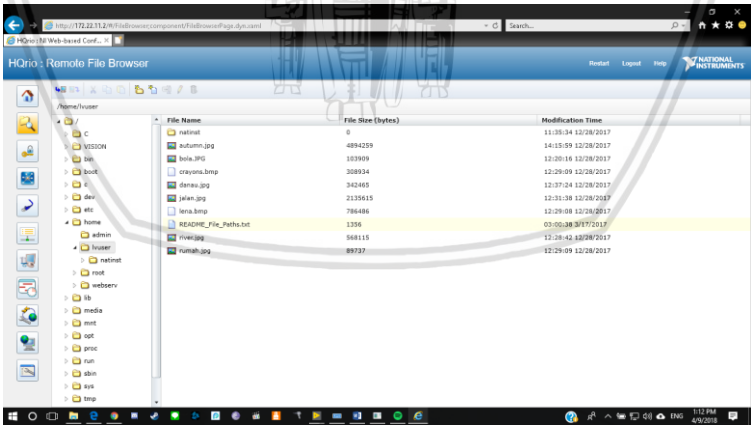
Koneksikan myRIO dengan PC menggunakan kabel USB dan koneksikan juga myRIO dengan Power Supply, lalu buka program LabVIEW dan pilih project untuk penelitian ini. Setelah itu buka VI pada sisi Host dan munculah jendela seperti Gambar 4.5



Gambar 4.5 Host VI pada program LabVIEW

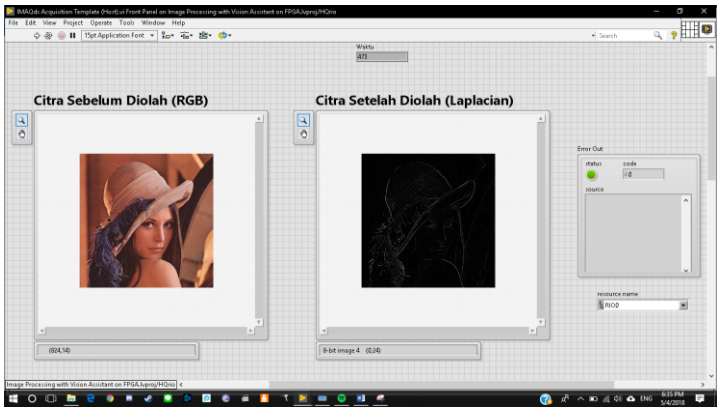
Sebelum menjalankan program, harus terlebih dahulu dipilih *Resource Name* pada *Front Panel* program. *Resource Name* digunakan untuk menentukan *device target* yang dipilih dan program akan dijalankan pada *device* tersebut. Setelah dipilih, baru bisa menjalankan program dengan mengklik tombol *Run*

Untuk memilih citra yang akan diolah, citra perlu dimasukan ke memori myRIO. Buka *Internet Explorer* yang tersedia pada OS Windows, lalu pada *address bar* input IP Address dari myRIO yang secara default adalah http://172.22.11.2/. Setelah itu akan muncul halaman *System Configuration* dari myRIO, lalu kita harus login untuk dapat mengakses *File Browser* pada myRIO untuk mengatur file seperti yang terlihat pada Gambar 4.6



Gambar 4.6 Halaman File Browser myRIO

Untuk mengganti file citra yang akan diolah, kita perlu mendeklarasikanya direktorinya pada VI *IMAQ ReadFile* setelah kita upload pada *File Browser*. Lalu citra dapat diolah dan hasilnya seperti pada Gambar 4.7

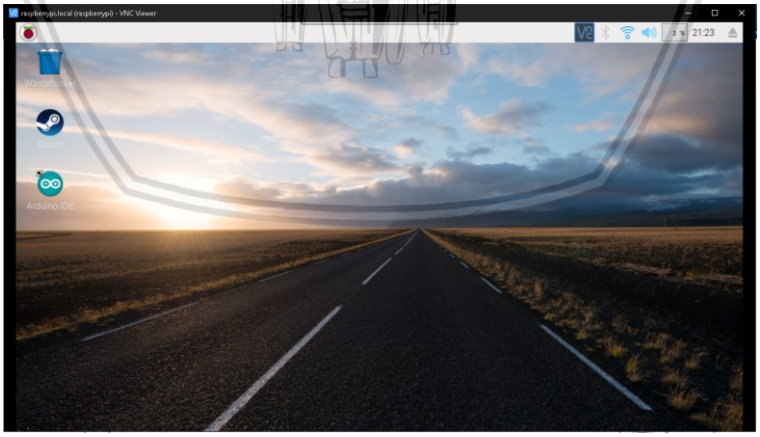


Gambar 4.7 Tampilan Front Panel LabVIEW setelah mengolah citra

4.4 Implementasi Eksekusi Sistem pada Mikrokomputer

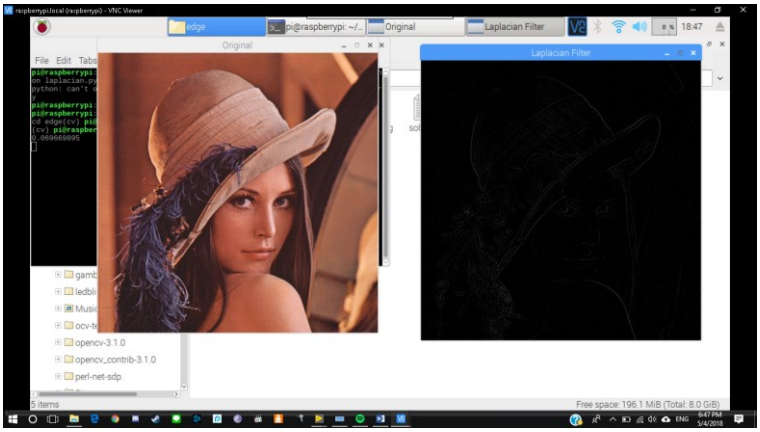
Pada tahap implementasi pengolah citra pada Mikrokomputer akan dijelaskan tahap-tahap yang dilakukan agar sistem dapat bekerja sesuai dengan yang diharapkan. Implementasi pengolah citra pada Raspberry Pi dilakukan dengan tahapan-tahapan sebagai berikut:

Pertama koneksikan Raspberry Pi ke PC menggunakan kabel LAN dan hubungkan juga *Power Supply*. Lalu buka program VNCViewer pada PC dan masukan alamat raspberrypi.local untuk mengakses Raspberry Pi secara local. Setelah dijalankan, lalu akan terkoneksi dengan Raspberry Pi dan bisa melihat tampilan *desktop* OS Raspbian seperti pada Gambar 4.8



Gambar 4.8 Desktop OS Raspbian

Lalu buka terminal yang ada pada Raspbian OS untuk mengakses *Environtment* OpenCV. Setelah mendapaatkan akses, lalu bisa menjalankan kode program dengan perintah *python laplacian.py* dan munculah citra sebelum diolah dan citra setelah diolah seperti yang bisa dilihat pada Gambar 4.9



Gambar 4.9 Program Mengolah Citra

BAB 5 PENGUJIAN DAN ANALISIS

Pada bab ini membahas mengenai pengujian dan analisis hasil dari implementasi sistem yang telah diterapkan melalui tahapan implementasi. Pengujian dilakukan untuk mengetahui apakah kebutuhan sistem telah terpenuhi. Pengujian dibagi menjadi beberapa tahap sesuai dengan tujuannya agar lebih mudah dalam proses analisis. Pengujian dilakukan dengan menghitung waktu awal sistem bekerja yaitu proses mengambil dan menampilkan citra sebelum diolah, proses mengolah citra dan proses mengambil dan menampilkan citra setelah diolah. Pengujian tidak dapat melakukan perhitungan waktu pemrosesan citra saja karena pada LabVIEW tidak mendukung fungsi *Tick Count* pada VI di FPGA.

5.1 Pengujian Pengolahan Citra dengan Algoritma *Gaussian Blur*

5.1.1 Tujuan Pengujian

Pengujian ini bertujuan untuk mengolah citra dari citra berwarna RGB menjadi citra *grayscale* yang telah dihaluskan dengan algoritma *Gaussian Blur*. Selain menghasilkan citra baru, pengujian ini bertujuan untuk mengetahui waktu pengolahan citra dengan algoritma *Gaussian Blur* pada sistem FPGA dan Mikrokomputer

5.1.2 Prosedur Pengujian

5.1.2.1 Prosedur Pengujian pada FPGA

Prosedur yang dilakukan pada pengujian ini adalah sebagai berikut :

1. Perangkat myRIO yang pada kondisi menyala dan telah terkoneksi dengan PC
2. Membuka aplikasi LabVIEW yang berisi program pengolahan citra dengan algortima *Gaussian Blur*
3. Menginisialilasi nama file citra yang akan diolah pada blok diagram program LabVIEW
4. Memilih *resource name* untuk menentukan *target device* dimana program akan dijalankan
5. Mengklik tombol *run* pada aplikasi LabVIEW

5.1.2.2 Prosedur Pengujian pada Mikrokomputer

Prosedur yang dilakukan pada pengujian ini adalah sebagai berikut :

1. Perangkat Raspberry Pi yang pada kondisi menyala dan telah terkoneksi dengan PC
2. Membuka Aplikasi VNCViewer untuk meremote Raspberry Pi
3. Menginisialilasi nama file citra yang akan diolah pada file program yang akan dijalankan
4. Membuka terminal untuk mengakses *environment* dari OpenCV
5. Menjalankan program dari terminal dengan perintah `python gaussian.py`

5.1.3 Hasil Pengujian

5.1.3.1 Hasil Pengujian pada FPGA

Hasil pengujian ini akan ditampilkan pada *front panel* dari aplikasi LabVIEW. Pada Gambar 5.1 adalah hasil pengujian dari citra rusa pada program LabVIEW dengan algoritma *Gaussian Blur.*

Gambar 5.1 Citra Rusa dengan Algoritma *Gaussian Blur* pada Aplikasi LabVIEW

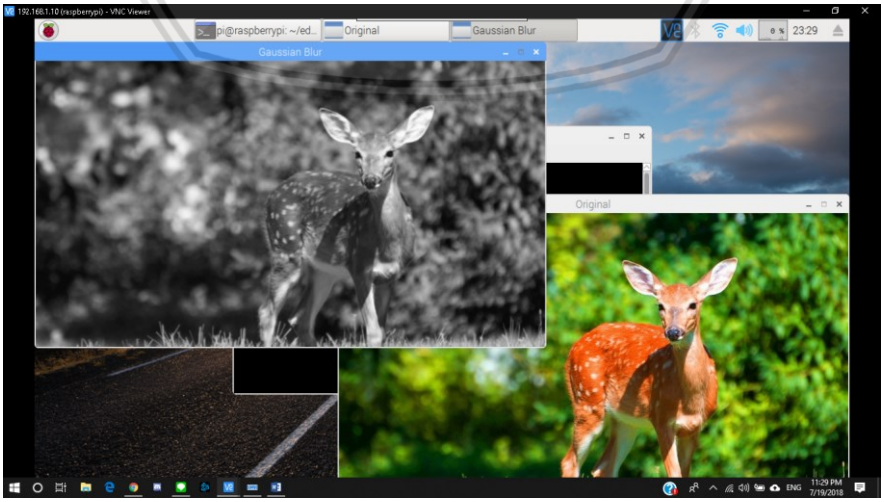
Pada tahap pengujian ini juga didapatkan nilai waktu proses pengolahan citra dengan algoritma *Gaussian Blur.* Pengujian dilakukan terhadap tiga citra yang berbeda resolusi yaitu kecil, sedang dan besar. Pengujian ini dilakukan sebanyak 10 kali dan hasil pengujian dapat dilihat pada Tabel 5.1:

Tabel 5.1 Hasil Percobaan Nilai Waktu pada Citra dengan Algoritma *Gaussian Blur* pada FPGA

Percobaan Ke-	Nilai Waktu (detik)		
Jenis Citra	Kecil (800x449)	Sedang (1366x768)	Besar (1920x1080)
1	0.178	0.445	0.810
2	0.183	0.465	0.821
3	0.180	0.457	0.812
4	0.176	0.454	0.824
5	0.175	0.455	0.823
6	0.178	0.453	0.821
7	0.173	0.456	0.832
8	0.177	0.460	0.825
9	0.172	0.455	0.822
10	0.179	0.459	0.833
Rata-rata	0.177	0.456	0.822

5.1.3.2 Hasil Pengujian pada Mikrokomputer

Hasil pengujian ini akan ditampilkan pada *desktop* dari OS Raspbian. Pada Gambar 5.2 adalah hasil pengujian dari citra rusa pada *desktop* dari OS Raspbian dengan algoritma *Gaussian Blur.*



Gambar 5.2 Citra Rusa dengan Algoritma *Gaussian Blur* pada Dekstop Raspbian

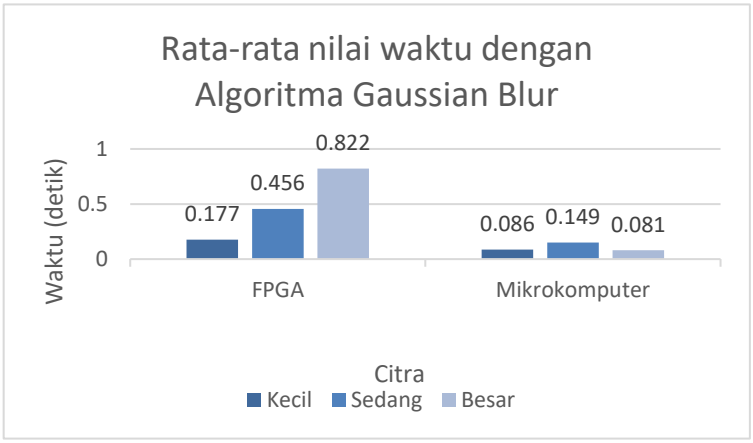
Pada tahap pengujian ini juga didapatkan nilai waktu proses pengolahan citra dengan algoritma *Gaussian Blur.* Pengujian dilakukan terhadap tiga citra yang berbeda resolusi yaitu kecil, sedang dan besar.Pengujian ini dilakukan sebanyak 10 kali dan hasil pengujian dapat dilihat pada Tabel 5.2 :

Tabel 5.2 Hasil Percobaan Nilai Waktu pada Citra dengan Algoritma *Gaussian Blur* pada Mikrokomputer

Percobaan Ke-	Nilai Waktu (detik)		
Jenis Citra	Kecil (800x449)	Sedang (1366x768)	Besar (1920x1080)
1	0.084	0.155	0.255
2	0.081	0.156	0.256
3	0.082	0.154	0.260
4	0.088	0.149	0.259
5	0.087	0.155	0.261
6	0.086	0.143	0.253
7	0.089	0.142	0.257
8	0.092	0.148	0.258
9	0.093	0.144	0.278
10	0.080	0.146	0.260
Rata-rata	0.086	0.149	0.260

5.1.4 Analisa Pengujian

Pada pengujian pengolahan citra dengan algoritma *gaussian blur*, sistem berhasil melakukan pengolahan citra dengan baik, grafik perbandingan hasil waktu pengolahan citra dapat dilihat pada Gambar 5.3:



Gambar 5.3 Grafik Rata-rata Nilai Waktu dengan Algoritma *Gaussian Blur*

5.2 Pengujian Pengolahan Citra dengan Algoritma *Laplacian Edge*

5.2.1 Tujuan Pengujian

Pengujian ini bertujuan untuk mengolah citra dari citra berwarna RGB menjadi citra *grayscale* yang telah dihaluskan dengan algoritma *Gaussian Blur*. Setelah citra dihaluskan, diterapkan algoritma *Laplacian Edge* untuk mendeteksi tepi pada citra. Selain menghasilkan citra baru, pengujian ini bertujuan untuk mengetahui waktu pengolahan citra dengan algoritma *Laplacian Edge* pada sistem FPGA dan Mikrokomputer.

5.2.2 Prosedur Pengujian

5.2.2.1 Prosedur Pengujian pada FPGA

Prosedur yang dilakukan pada pengujian ini adalah sebagai berikut :

1. Perangkat myRIO yang pada kondisi menyala dan telah terkoneksi dengan PC
2. Membuka aplikasi LabVIEW yang berisi program pengolahan citra dengan algortima *Laplacian Edge*
3. Menginisialilasi nama file citra yang akan diolah pada blok diagram program LabVIEW
4. Memilih *resource name* untuk menentukan *target device* dimana program akan dijalankan
5. Mengklik tombol *run* pada aplikasi LabVIEW

5.2.2.2 Prosedur Pengujian pada Mikrokomputer

Prosedur yang dilakukan pada pengujian ini adalah sebagai berikut :

1. Perangkat Raspberry Pi yang pada kondisi menyala dan telah terkoneksi dengan PC
2. Membuka Aplikasi VNCViewer untuk meremote Raspberry Pi
3. Menginisialilasi nama file citra yang akan diolah pada file program yang akan dijalankan

4. Membuka terminal untuk mengakses *environment* dari OpenCV
5. Menjalankan program dari terminal dengan perintah `python laplacian.py`

5.2.3 Hasil Pengujian

5.2.3.1 Hasil Pengujian pada FPGA

Hasil pengujian ini akan ditampilkan pada *front panel* dari aplikasi LabVIEW. Pada Gambar 5.4 adalah hasil pengujian dari citra rusa pada program LabVIEW dengan algoritma *Laplacian Edge*.

Gambar 5.4 Citra Rusa dengan Algoritma *Laplacian Edge* pada Aplikasi LabVIEW

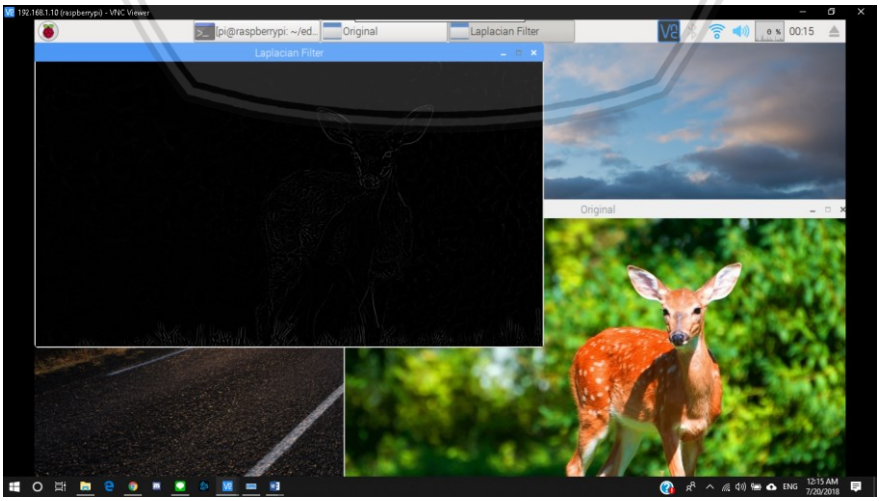
Pada tahap pengujian ini juga didapatkan nilai waktu proses pengolahan citra dengan algoritma *Laplacian Edge.* Pengujian dilakukan terhadap tiga citra yang berbeda resolusi yaitu kecil, sedang dan besar. Pengujian ini dilakukan sebanyak 10 kali dan hasil pengujian dapat dilihat pada Tabel 5.3:

Tabel 5.3 Hasil Percobaan Nilai Waktu pada Citra dengan Algoritma *Laplacian Edge* pada FPGA

Percobaan Ke-	Nilai Waktu (detik)		
Jenis Citra	Kecil (800x449)	Sedang (1366x768)	Besar (1920x1080)
1	0.177	0.457	0.812
2	0.181	0.455	0.821
3	0.176	0.470	0.820
4	0.177	0.465	0.828
5	0.175	0.463	0.813
6	0.180	0.466	0.816
7	0.178	0.468	0.823
8	0.177	0.455	0.815
9	0.181	0.458	0.813
10	0.177	0.456	0.813
Rata-rata	0.178	0.461	0.817

5.2.3.2 Hasil Pengujian pada Mikrokomputer

Hasil pengujian ini akan ditampilkan pada *desktop* dari OS Raspbian. Pada Gambar 5.5 adalah hasil pengujian dari citra rusa pada OS Raspbian dengan algoritma *Lapalacian Edge*.



Gambar 5.5 Citra Rusa dengan Algoritma *Laplacian Edge* pada Desktop Raspbian

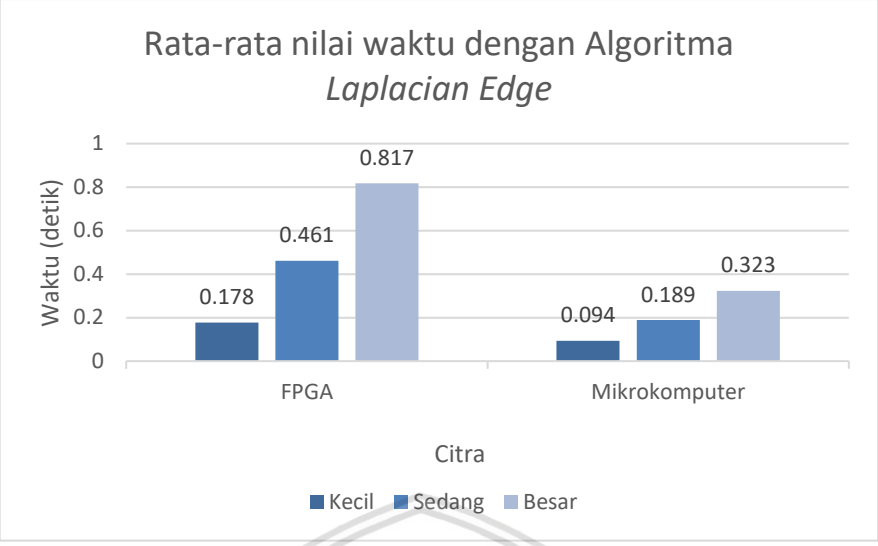
Pada tahap pengujian ini juga didapatkan nilai waktu proses pengolahan citra dengan algoritma *Laplacian Edge.* Pengujian dilakukan terhadap tiga citra yang berbeda resolusi yaitu kecil, sedang dan besar. Pengujian ini dilakukan sebanyak 10 kali dan hasil pengujian dapat dilihat pada Tabel 5.4:

Tabel 5.4 Hasil Percobaan Nilai Waktu pada Citra dengan Algoritma *Laplacian Edge* pada Mikrokomputer

Percobaan Ke-	Nilai Waktu (detik)		
Jenis Citra	Kecil (800x449)	Sedang (1366x768)	Besar (1920x1080)
1	0.097	0.188	0.320
2	0.094	0.189	0.322
3	0.094	0.180	0.329
4	0.091	0.189	0.321
5	0.093	0.188	0.321
6	0.091	0.180	0.324
7	0.095	0.187	0.320
8	0.096	0.188	0.322
9	0.093	0.188	0.320
10	0.093	0.180	0.329
Rata-rata	0.094	0.189	0.323

5.2.4 Analisa Pengujian

Pada pengujian pengolahan citra dengan algoritma *laplacian edge*, sistem berhasil melakukan pengolahan citra dengan baik, grafik perbandingan hasil waktu pengolahan citra dapat dilihat pada Gambar 5.6 :



Gambar 5.6 Grafik Rata-rata Nilai Waktu dengan Algoritma *Laplacian Edge*

5.3 Pengujian Pengolahan Citra dengan Algoritma *Sobel Edge*

5.3.1 Tujuan Pengujian

Pengujian ini bertujuan untuk mengolah citra dari citra berwarna RGB menjadi citra *grayscale* yang telah dihaluskan dengan algoritma *Gaussian Blur*. Setelah citra dihaluskan, diterapkan algoritma *Sobel Edge* untuk mendeteksi tepi pada citra. Selain menghasilkan citra baru, pengujian ini bertujuan untuk mengetahui waktu pengolahan citra dengan algoritma *Sobel Edge* pada sistem FPGA dan Mikrokomputer

5.3.2 Prosedur Pengujian

5.3.2.1 Prosedur Pengujian pada FPGA

Prosedur yang dilakukan pada pengujian ini adalah sebagai berikut :

1. Perangkat myRIO yang pada kondisi menyala dan telah terkoneksi dengan PC
2. Membuka aplikasi LabVIEW yang berisi program pengolahan citra dengan algortima *Sobel Edge*
3. Menginisialilasi nama file citra yang akan diolah pada blok diagram program LabVIEW
4. Memilih *resource name* untuk menentukan *target device* dimana program akan dijalankan
5. Mengklik tombol *run* pada aplikasi LabVIEW

5.3.2.2 Prosedur Pengujian pada Mikrokomputer

Prosedur yang dilakukan pada pengujian ini adalah sebagai berikut :

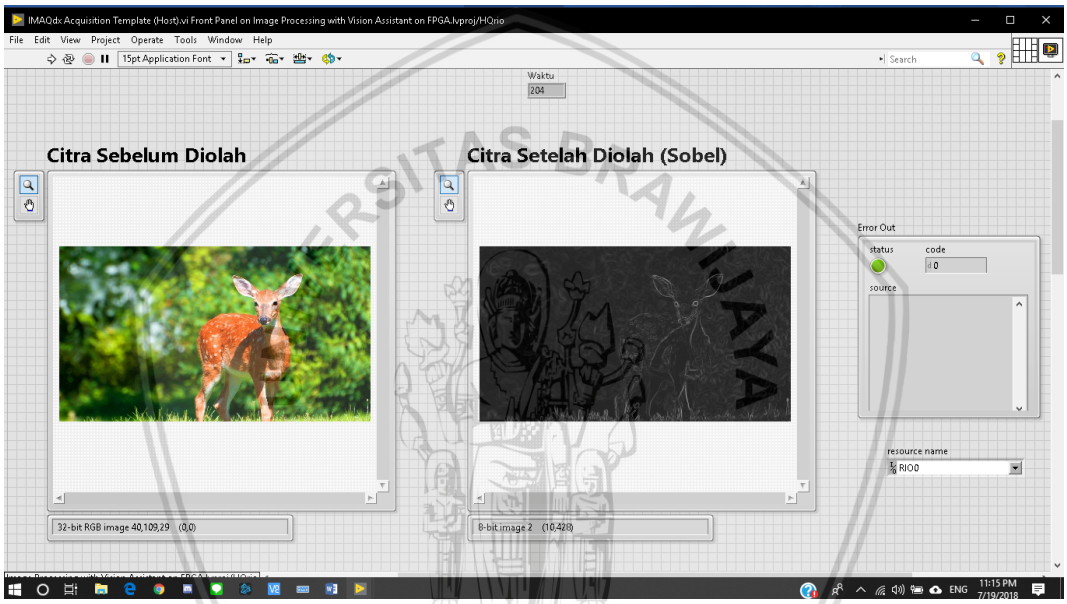
1. Perangkat Raspberry Pi yang pada kondisi menyala dan telah terkoneksi dengan PC
2. Membuka Aplikasi VNCViewer untuk meremote Raspberry Pi

3. Menginisialilasi nama file citra yang akan diolah pada file program yang akan dijalankan
4. Membuka terminal untuk mengakses *environment* dari OpenCV
5. Menjalankan program dari terminal dengan perintah `python sobel.py`

5.3.3 Hasil Pengujian

5.3.3.1 Hasil Pengujian pada FPGA

Hasil pengujian ini akan ditampilkan pada *front panel* dari aplikasi LabVIEW. Pada Gambar 5.7 adalah hasil pengujian dari citra rusa pada program LabVIEW dengan algoritma *Sobel Edge*.



Gambar 5.7 Citra Rusa dengan Algoritma *Sobel Edge* pada Aplikasi LabVIEW

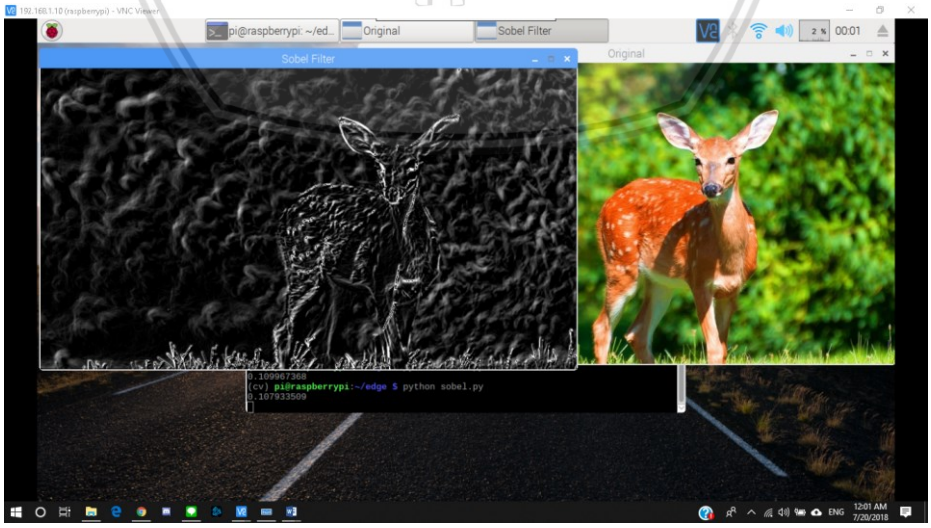
Pada tahap pengujian ini juga didapatkan nilai waktu proses pengolahan citra dengan algoritma *Sobel edge.* Pengujian dilakukan terhadap tiga citra yang berbeda resolusi yaitu kecil, sedang dan besar. Pengujian ini dilakukan sebanyak 10 kali dan hasil pengujian dapat dilihat pada Tabel 5.5 :

Tabel 5.5 Hasil Percobaan Nilai Waktu pada Citra dengan Algoritma *Sobel Edge* pada FPGA

Percobaan Ke-	Nilai Waktu (detik)		
Jenis Citra	Kecil (800x449)	Sedang (1366x768)	Besar (1920x1080)
1	0.180	0.460	0.831
2	0.184	0.470	0.823
3	0.186	0.469	0.834
4	0.179	0.465	0.828
5	0.182	0.459	0.830
6	0.184	0.467	0.829
7	0.185	0.463	0.826
8	0.180	0.462	0.827
9	0.176	0.466	0.820
10	0.184	0.454	0.828
Rata-rata	0.182	0.464	0.828

5.3.3.2 Hasil Pengujian pada Mikrokomputer

Hasil pengujian ini akan ditampilkan pada *desktop* dari OS Raspbian. Pada Gambar 5.8 adalah hasil pengujian dari citra crayon pada OS Raspbian.



Gambar 5.8 Citra Rusa dengan Algoritma *Sobel Edge* pada Desktop Raspbian

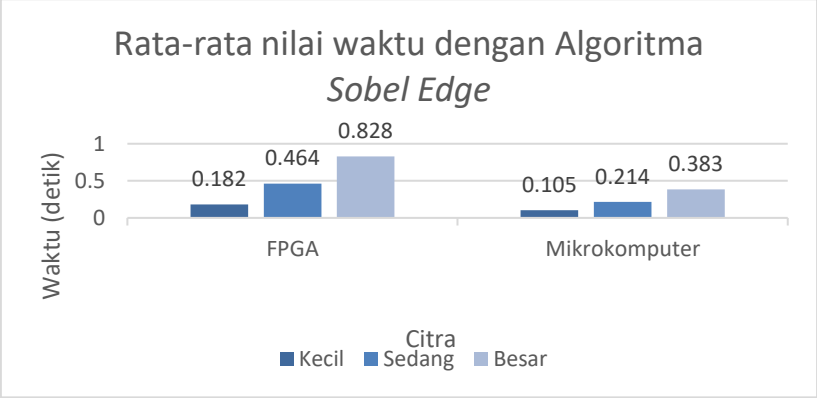
Pada tahap pengujian ini juga didapatkan nilai waktu proses pengolahan citra dengan algoritma *sobel edge.* Pengujian dilakukan terhadap tiga citra yang berbeda resolusi yaitu kecil, sedang dan besar. Pengujian ini dilakukan sebanyak 10 kali dan hasil pengujian dapat dilihat pada Tabel 5.6:

Tabel 5.6 Hasil Percobaan Nilai Waktu pada Citra dengan Algoritma *Sobel Edge* pada Mikrokomputer

Percobaan Ke-	Nilai Waktu (detik)		
Jenis Citra	Kecil (800x449)	Sedang (1366x768)	Besar (1920x1080)
1	0.107	0.214	0.382
2	0.108	0.215	0.382
3	0.105	0.213	0.387
4	0.108	0.215	0.381
5	0.108	0.215	0.380
6	0.104	0.216	0.383
7	0.105	0.211	0.382
8	0.100	0.213	0.384
9	0.105	0.214	0.381
10	0.102	0.213	0.385
Rata-rata	0.105	0.214	0.383

5.3.4 Analisa Pengujian

Pada pengujian pengolahan citra dengan algoritma *sobel edge*, sistem berhasil melakukan pengolahan citra dengan baik, grafik perbandingan hasil waktu pengolahan citra dapat dilihat pada Gambar 5.9:

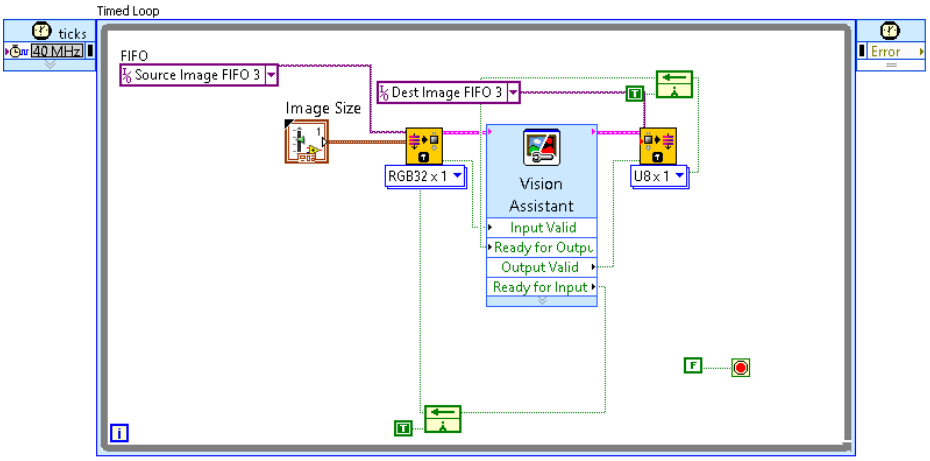


Gambar 5.9 Grafik Rata-rata Nilai Waktu dengan Algoritma *Sobel Edge*

5.4 Analisis Hasil Keseluruhan Pengujian

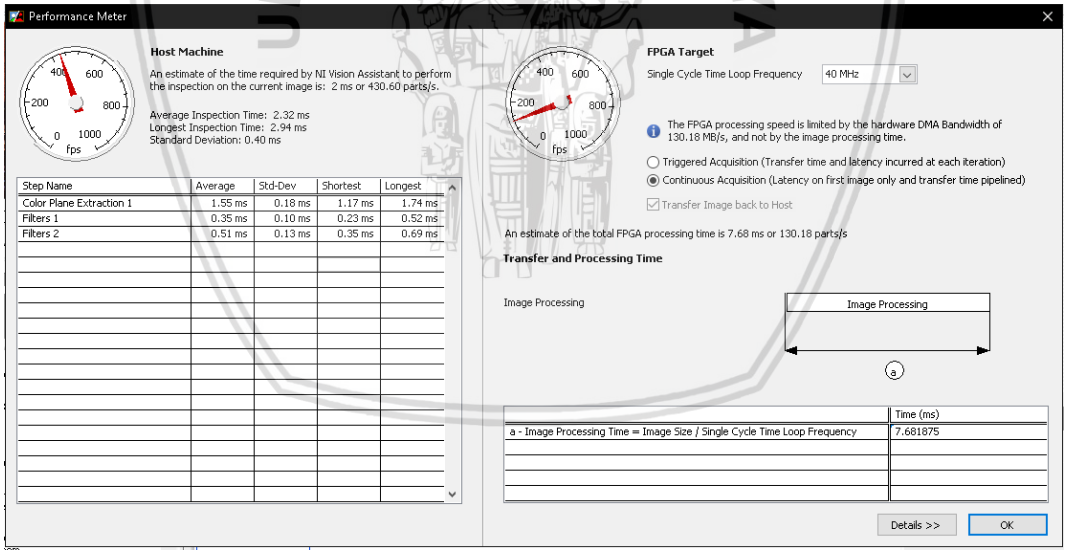
Berdasarkan hasil pengujian dapat dikatakan sistem dapat dengan baik mengolah citra, namun ada perbedaan nilai waktu pada kedua *platform* yaitu di FPGA maupun Mikrokomputer. Hasil waktu rata-rata pengolahan citra dengan algoritma *Gaussian blur* adalah 0.485s pada FPGA dan 0.165s pada Mikrokomputer. Untuk algortitma *Laplacian edge* waktu rata-rata adalah 0.492s pada FPGA dan 0.202s pada Mikrokomputer sedangkan untuk algoritma *Sobel edge* waktu rata-rata nya adalah 0.498s pada FPGA dan 0.234s pada Mikrokomputer.

Hasil pengujian membutikkan, bahwa Mikrokomputer lebih unggul dalam kecepatan keseluruhan pengolahan citra dibandingkan FPGA pada beberapa algoritma dan jenis citra. Hal ini disebabkan karena pada blok diagram LabVIEW, yang dihitung adalah kecepatan transfer data citra dari Host ke FPGA sebelum citra diolah dan transfer data citra dari FPGA ke Host setelah citra diolah. Bisa dilihat dari data nilai rata-rata waktu FPGA untuk citra yang sama dengan algoritma yang berbeda, waktunya juga tidak jauh berbeda. Untuk waktu pengolahan citra, pada blok diagram LabVIEW di sisi VI FPGA digunakan *Single Cycle Timed Loop.* Pada Gambar 5.10 adalah blok diagram fungsi pengolahan citra yang menggunakan *Single Cycle Timed Loop.*



Gambar 5.10 *Single Cycle Timed Loop* pada Fungsi Pengolahan Citra

Single Cycle Timed Loop ini berfungsi untuk memastikan setiap fungsi dan algoritma yang ada di dalam loop ini tereksekusi selama satu *cycle* pada *clock rate* yang kita tentukan.Didalam Loop ini terdapat blok diagram dari VI *Vision Assistant,* yang dimana jika di *compile* maka akan otomatis memparalellisasi code dihasilkan. MyRIO memiliki *clock rate* bawaan sebesar 40 MHz yang berarti memiliki clock cycle sebesar 25 ns. Sedangkan kecepatan mengolah citra pada FPGA berbeda bergantung pada citra yang akan diolah, kecepetan mengolah citra pada FPGA bisa di estimasi dengan Tools yang ada di VI *Vision Assistant* yaitu tools *Perfomance Meter*. Tampilan Perfomance Meter pada VI Vision Assistant dapat dilihat pada Gambar 5.11 berikut ini.



Gambar 5.11 *Perfomance Meter* pada *Vision Assistant*

Hasil estimasi waktu pengolahan citra pada FPGA menunjukan, waktu pengolahan pada citra berukuran kecil adalah 0.01053 detik, citra berukuran sedang adalah 0.03074 detik dan citra berkuran besar adalah 0.06076 detik. Hasil kecepatan ini dipengaruhi oleh karakteristik dari FPGA sendiri yang memiliki kemampuan pemrosesan parallel dan logika yang dibuat bekerja secara kombinatorial pada level perangkat keras.

Sedangkan hasil pengujian pada Mikrokomputer Raspberry Pi menunjukan waktu pengolahan yang lebih cepat dibandingkan FPGA. Namun pengolahan citra

pada Raspberry Pi dilakukan secara sekuensial pada level perangkat lunak, tidak seperti pengolahan citra pada FPGA yang dilakukan secara parallel pada level perangkat keras. Ini dibuktikan dengan hasil pengujian pada Raspberry Pi, algoritma *Gaussian Blur* memiliki waktu pengolahan lebih cepat dibandingkan algoritma *Laplacian Edge* dan *Sobel Edge* karena algoritma *Gaussian Blur* lebih sederhana dibandingkan kedua algoritma lainnya. Begitu juga dengan hasil dari pengolahan citra dengan algoritma *Laplacian Edge* lebih cepat daripada algoritma *Sobel Edge*, ini disebabkan algoritma *Sobel Edge* memiliki aritmatika lebih komples dalam perhitungannya dibandingkan dengan algoritma *Laplacian Edge.*

BAB 6 PENUTUP

Bagian ini memuat kesimpulan dan saran terhadap skripsi. Kesimpulan dan saran disajikan secara terpisah, dengan penjelasan sebagai berikut:

6.1 Kesimpulan

Pada bab ini akan menjawab pertanyaan dari rumusan masalah yang digunakan sebagai kesimpulan yang diambil pada penulisan tugas akhir ini. Skenario yang digunakan dalam pengujian menggunakan algoritma *Gaussian Blur, Laplacian Edge* dan *Sobel Edge* adalah dengan mengolah tiga citra yang berbeda dan melakukan pengolahan citra sebanyak 10 kali dan diambil nilai waktunya. Pada pengujian menggunakan algoritma pengolahan citra yang telah diuji dapat ditunjukkan hasil berikut :

1. Sistem pengolahan citra dapat dengan algoritma *Gaussian Blur, Laplacian Edge* dan *Sobel Edge* pada platform FPGA dapat di implementasikan menggunakan aplikasi LabVIEW dan Library dari NI Vision Assistant, sedangkan pada platform FPGA algoritma tersebut dapat dirancang dengan OS Raspbian dan Library dari OpenCV
2. Pada pengujian nilai waktu pengolahan citra, rata-rata waktu pada tiga algoritma menggunakan tiga citra yang berbeda selama 10 kali dengan algoritma *Gaussian blur* adalah 0.485s pada FPGA dan 0.165s pada Mikrokomputer. Untuk algortitma *Laplacian edge* waktu rata-rata adalah 0.492s pada FPGA dan 0.202s pada Mikrokomputer sedangkan untuk algoritma *Sobel edge* waktu rata-rata nya adalah 0.498s pada FPGA dan 0.234s pada Mikrokomputer. Namun sebenarnya untuk semua algoritma, waktu FPGA tetap sama namun berbeda pada tiap citra berukuran kecil, sedang dan besar masing-masing, adalah 0.01053 detik, 0.03074 detik dan 0.06076 detik.
3. Waktu akuisisi citra sebelum dan setelah citra diolah di Mikrokomputer memang lebih unggul dibandingkan FPGA, namun untuk waktu pengolahan citra FPGA jauh lebih unggul dikarenakan karakteristik FPGA yang mampu bekerja secaral parallel dan logika yang dibuat bekerja secara kombinatorial pada level perangkat keras
4. Hipotesis tentang pengaruh pengaruh kecepatan pemrosesan FPGA yang memiliki sifat pemrosesan di implementasikan hingga tataran perangkat keras lebih cepat waktu pemrosesannya dibandingankan Mikrokomputer dapat dibuktikan dengan waktu pemrosesan citra pada FPGA tanpa dihitung waktu akuisisi citra lebih cepat dibandingkan Mikrokomputer

6.2 Saran

Penulisan tugas akhir ini tentunya masih ada beberapa kekurangan yang masih belum dapat dikerjakan dengan baik. Saran yang dapat dilakukan pada penelitian selanjutnya adalah sebagai berikut :

Pengujian menggunakan device lain pada platform yang sama agar hasil pengujian terlihat perbedannya

2. Skenario pengujian menggunakan metode lain seperti *Canny Edge* atau *Thresholding*
3. Untuk peneletian lebih lanjut, bisa menggunakan video untuk menguji performa sistem

DAFTAR PUSTAKA

Abda'i, H. (2016). NOISE REDUCTION PADA CITRA DIGITAL DENGAN METODE GAUSSIAN FILTER . *Sekolah Tinggi Teknik Harapan Medan*.

Ahmad, U. (2005). *Pengolahan Citra Digital & Teknik Pemrogramannya.* Bogor: Graha Ilmu.

Kiran, M., War, K. M., Kuan, L. M., Meng, L. K., & Kin, L. W. (2008). Implementing image processing algorithms using 'Hardware in the loop' approach for Xilinx FPGA. *International Conference on Electronic Design*.

Maharani, D. S., Apriyana, Puspasari, S., & Angreni, R. (2007). Perbandingan Metode Sobel, Metode Prewitt dan Metode Robert Untuk Deteksi Tepi Objek Pada Aplikasi Pengenalan Bentuk Berbasis Citra Digital . *STMIK GI MDP*.

Mulya, M., & Abdiansah. (2013). Penerapan Multi-threading untuk Meningkatkan Kinerja Pengolahan Citra Digital. *Universitas Sriwijaya*.

Najihi, A. (2016). ANALISIS KINERJA IP PBX SERVER PADA SINGLE BOARD CIRCUIT RASPBERRY PI. *Universitas Muhammadiyah Palangkaraya*.

OpenCV. (2012, August 23). *OpenCV Library*. Retrieved from OpenCV Web Site: https://opencv.org/

Prasetiyo, A. F. (2015). Penerapan Gaussian Filter pada Edge Detection . *Institut Teknologi Bandung*.

Proakis, J. G. (2005). *Digital Signal Processing.* New Jersey: Prentice-Hall Inc.

Sholihin, R. A., & Purwoto, B. H. (2015). PERBAIKAN CITRA DENGAN MENGGUNAKAN MEDIAN FILTER dan METODE HISTOGRAM EQUALIZATION. *Universitas Muhammadiyah Surakarta*.

Xilinx. (2017). *Zynq-7000 All Programmable SoC Product Advantages*. Retrieved 2018, from https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html

Yuwono, B., Nugroho, S. P., & Herlyanto. (n.d.). Pengembangan Model Public Monitoring System menggunakan Raspberry Pi. *Jurnal Telematika*, 2015.

Zuhdy, A. H. (2014). Implementasi Programmable DAC pada FPGA Xilink Spartan-6 Berbasis VHDL. *Universitas Gajah Mada*.